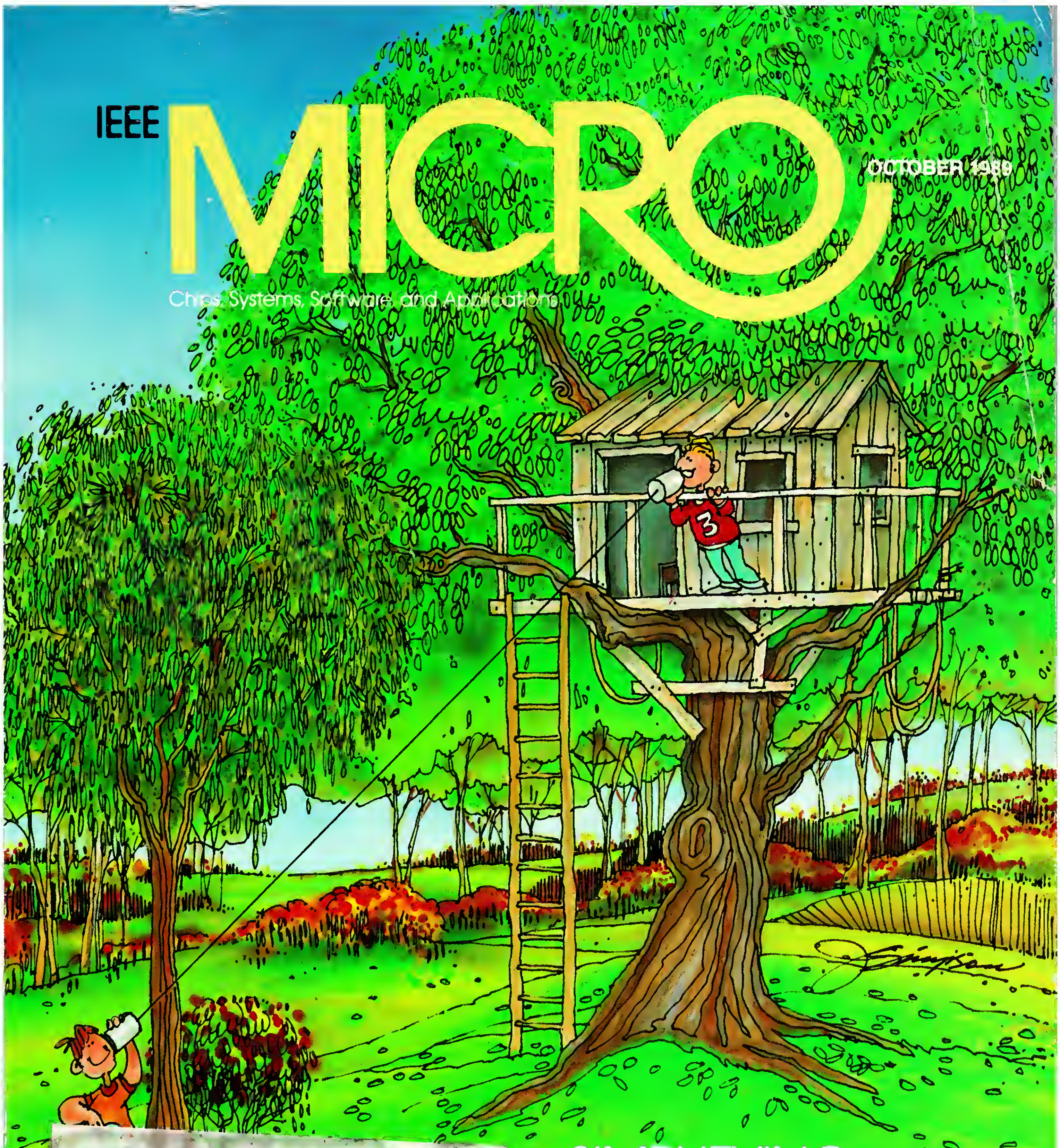
IEEE MICRO

Chips, Systems, Software, and Applications

OCTOBER 1989

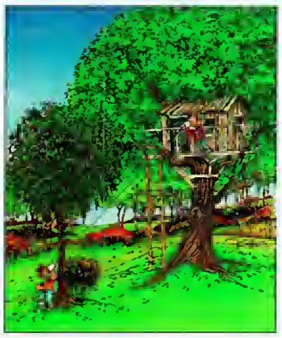


SIMPLIFYING INTERPROCESSOR COMMUNICATION

- Multiprocessor hardware monitor
- Using the fast Hartley transform
- VLSI neural network chip
- and more

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.


Cover by Jay Simpson, Design & Direction

IEEE MICRO

Volume 9 Number 5 (ISSN 0272-1732) October 1989

Published by the IEEE Computer Society

Departments

Special Features

The Intel Influence

FOCUSED. DRIVEN. MOTIVATED.

i486™ BRAND MICROPROCESSOR

It's Intel. And the high achieving individuals who thrive and succeed in our energy-charged atmosphere. But the results have paid off, with one innovation after another. Intel has become much more than just another semiconductor company...a leading microcomputer company offering opportunities at components and systems levels.

Changing history through innovations in the industry has long been Intel's strength. We are currently seeking professionals for the following areas:

- **Operating Systems**
- **Systems Architecture**
- **Architecture Simulation**
- **I/O Subsystems (disk, network, etc.)**
- **Compilers**
- **Benchmarking**
- **Competitive Analysis**
- **Technical Marketing**
- **Hardware Systems Design**
- **Multiprocessing**
- **PCAT-Architecture**
- **VLSI-Logic/Circuit Design**

We also have positions available for Senior Technical Staff Members in the following areas:

- **System Architecture and future directions.**
- **CPU/Microprocessors**
- **I/O Architectures and future components.**
- **Plan, manage, implement and operate a lab for benchmarking systems from Intel and other vendors.**
- **Manage a small group that will develop a strategy and drive implementation of tools to simulate complete/partial systems.**

VLSI Design - Logic & Circuit

Requires 3 plus years industrial experience on microprocessors, peripherals or memories.

Compiler Software Engineer

Responsible for generating compiler code and utilizing experience with RISC architectures.

Technical Marketing Engineers

Will target customers for the i860™ microprocessor or i486™ microprocessor. Requires workstation, minicomputer, graphics systems, or PCAT architecture expertise.

All positions require extensive experience and a graduate degree, MSEE or CS required.

If you'd like to learn more about the positives of being focused, driven, motivated, we'd like to hear from you. For immediate consideration, please call Rebecca at (408) 765-1546 or send your resume to: **Intel Corporation, Dept. S396, P.O. Box 58119, Santa Clara, CA 95052-8119.**
An EEO employer actively seeking m/f/v/h candidates.

intel®

IEEE Computer Society
PO Box 3014
Los Alamitos, CA 90720-1264
(714) 821–8380

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: $18 in addition to IEEE Computer Society or any other IEEE society member dues; $33 for members of other technical organizations. Back issues: $10 for members; $20 for nonmembers. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

IEEE Micro

* Submit six copies of all articles and special-issue proposals to Joe Hootman, EE Dept., University of North Dakota, PO Box 7165, Grand Forks, ND 58202; (701) 777-4331 Compmail+ j.hootman

From the Editor-in-Chief

This issue

This October issue of *IEEE Micro* discusses a wide variety of topics. While it carries no central theme, future issues will. Our European edition this December will present neural network articles concerning the current research on this form of parallel computing. The February 1990 issue will be dedicated to several of the best presentations given at the recently held Hot Chips Conference at Stanford University.

The first article in the current issue presents N. Jagadish, J. Mohan Kumar, and L.M. Patnaik discussing a procedure that uses dual-ported RAMs to help multiple processors communicate. The scheme presented has a large bandwidth and minimal software and hardware costs as well as being reliable. Next, Tho Le-Ngoc and Minh Tue Vo discuss implementation of the fast Hartley transform, which deals efficiently with real data. After evaluating FHT execution times and memory requirements, the authors compare these parameters with the fast Fourier transform for both hardware and software implementations.

Author Walt Price discusses benchmark tools, their use, and the results of using the benchmarks on various well-known computer systems. An-Chi Liu and Ranjani Parthasarathi look at the problem of monitoring a multiple microprocessor system with available hardware. Monitoring the performance of complex systems is important because performance can be tuned to make the system more efficient and cost effective. Monitoring a multiple processor system also tells the user the overall performance characteristics of the system, thus allowing the system to be configured in an optimal manner.

David Franklin and David Ostler present their work with the proposed IEEE standard (IEEE P1073) for interfacing medical devices to a computer. P1073 differs from most interfacing and local area network standards due to the environment in which the medical instrumentation must function.

Because computer design, development, and synthesis are near and dear to most *Micro* readers' hearts, we offer an article on the subject by William Birmingham, Anurag Gupta, and Daniel Siewiorek. The Micon system synthesizes computer systems from higher level specifications. The Micon set of programs supports hardware system design. Its very sophisticated software makes use of artificial intelligence to achieve the desired synthesis.

We add to the considerable amount of research and development work going on in the area of neural networks with our article by Mark Walker, Paul Hassler, and Lex Akers. These authors discuss the problems encountered in designing neural networks and the methods they chose to circumvent these problems. Their design of an adaptive CMOS neural array proves interesting.

Also with this issue Carl Warren joins the Editorial Board of *IEEE Micro*. Carl works for McDonnell-Douglas and has been active for many years in the IEEE

In the mailbag

December

"I liked the DSP articles, very readable indeed." W.F.D., Southampton, England

"I would like to see in-depth articles on the field bus, Bitbus, CAN, Profibus, MIL-1553, FIP, and ISI bus." R.T.S., Ranchi, India (As I mentioned earlier, we will soon be adding a Micro Standards department written by an editor with an interest in standards and buses. He will be on board soon.—J.H.)

"I liked the total issue. That was an excellent presentation of DSP. I would like to see [articles on] buses (Futurebus) and related products (special chips and boards)." B.C., Zagreb, Yugoslavia

"I liked the whole magazine. Good job." M.S.S., CAFB, MS

"I would like to see Bitbus, MIL-1553, Mini-Map, and CAN field buses compared, and their supporting chips." P.R., Tampere, Finland

"I liked Micro Review. I would like to see more materials on commercial software and hardware." F.F., Tehran

February

"I liked the robot-arm controller article. I would like to see more on commercial software packages." F.F., Tehran

"I would like to see articles on the i860, Next, and i486." V.R.M., Bangalore, India (A detailed article on the i860 appeared in the August issue of *Micro*, and we briefly discussed the Next system in June's Micro View column. We hope to have an article on the 486 in the February Hot Chips issue.—J.H.)

"I liked the Prolog coprocessor, the MARS approach, and EMMA2, A High-Performance Hierarchical Multiprocessor." U.S.S., Lima

"I liked virtually all the content. I would like to see articles on microprocessor applications to command and control systems and more on robotics." M.E.A., Jeddah, Saudi Arabia

April

"I would like to see more simple ideas on hardwired-logic simulation and industrial controllers." L.P., Recife, Brazil

"I liked the Micro Law article. Articles like this are very informative." R.C., Chittenango, NY

"I would like to see more of those technical reviews (Motorola 88000). Why [present] only the microprocessor and not the MMU (memory management)?" M.L., Grimbergen, Belgium (We have a manuscript in the review process that covers external MMU chips. We hope to have the review completed shortly.—J.H.)

"I liked the article on the Motorola 88000. I would like to see an article on the Intel N-10." A.L.B., Bulacan, Philippines (See our August issue; Intel now calls it the i860.—J.H.)

"See the interest codes! How about [supplying] pricing information on microprocessors reviewed? The editor says that a database management system is a software tool for microprocessor design. I do not agree!" P.S., Brentwood, NY (The interest codes found at the end of each article and department in every issue are important to us for two reasons. We want to know that you have read the article, and we want to know what you think about what you have read. This data helps us select the areas on which we should concentrate when considering the future editorial content of *Micro*.—J.H.)

June

"I would like to see some articles on network model databases. Relational is over emphasized. It's time to shift the focus." S.W., Sherbrooke, Canada

"I would like to see [articles on] commercial LAN-based products." T.I., Rochester, NY

"I liked the special issue on DSP [December 1988] very much. I would like to see implementations for DSP algorithms by using the transputers." P.T., Pittsburgh

"I liked Letters, Reviews. I disliked articles on TRON." S.M., North York, Canada

standards committees and in the technical publications business. He plans to revive our Micro Standards department in the near future.

Three members of the board—Dave Gustavson, Ken Majithia, and Shmuel Ben-Yaakov—leave the board. These three men have served *Micro* for four years (two terms) and cannot be reappointed. Dave Gustavson has been one of our prime article reviewers, contributing many ideas and supporting many of the special issues that we have published. Ken Majithia, another excellent reviewer, also served in the past as our New Products editor. Shmuel Ben-Yaakov provided reviews for the last four years. His comments and selection of other reviewers enhanced the quality of the articles and the magazine. I wish to thank these three men for their contributions to the excellence of *Micro* and wish them the best. They will be missed.

Just recently, I was discussing *Micro* with one of its founders. He pointed out that the original concepts for the magazine included the exchange of software and ideas on the efficient use of existing software. Since I have been Editor-in-Chief, I have seen several full-length articles on software and its uses. However, I have not seen any short (letter-length) software programs that might be of interest to readers. If you have a short program that you think might interest the readers of *Micro*, why not send it in?

J. Hootman

Micro World

Hubert Kirrmann
Asea Brown Boveri Research Center
CRBC.1
CH-5405 Baden, Switzerland

Jessi: A project to answer the giants

In the last few years, Europe has exploded with government-sponsored programs for the development of high technology. Every firm, university, or research institute tries to get its share of European Counting Units, or ECUs, from a European economic community program such as ESPRIT, Eureka, and Race (see Table 1).

Europeans originally regarded these programs as answers to US Department of Defense projects or to Japan's Fifth Generation Computer project sponsored by MITI, the Ministry of International Trade and Industry. All of these programs rely on the belief that if you just give scientists enough money and facilities they will come up with productive inventions to boost the country's national gross product.

In practice, things look a little different. European community money does not flow liberally. ESPRIT grants do not exceed 50 percent of project costs—and they are usually much lower. For Eureka (Mitterand's European Research Initiative), the Brussels administrative office pays solely for meetings. All programs require that firms and research institutes

Table 1.
European programs in information and communication technology.

Name	Topics	No. of Projects	Budget in ECUs* (millions)
Aim (1988)	Application of informatics to medicine	43	20
Cost (1971)	Telecommunications, transportation, oceanography, new materials, environmental protection, meteorology, agriculture	37	250
Delta (1988)	Application of informatics to teaching	30	20
Drive (1988)	Traffic control, automotive electronics	61	60
ESPRIT I (1983)	Microelectronics, software engineering, office automation, computer-integrated manufacturing	226	750
ESPRIT II (1987)	Continuation of ESPRIT I, basic research, artificial intelligence, computer science	158	1,600
Eureka (1985)	Informatics, telecommunication, computer-integrated manufacturing, robotics, new materials, lasers, biological techniques, environmental protection, energy, and transportation	297	6,400
Jessi (1987)	Microelectronics, new technologies, applications, manufacturing, basic research	—	3,800
Race (1987)	Technological support for telecommunications	94	550

*An ECU is worth about $1 in US currency.

collaborate towards a common goal. This approach requires a steering committee and kilometers of red tape. Collaboration within a group is shadowed by suspicions that other members will benefit more from the total outcome. (If it really *is* a profitable research subject, why should we share it with *them*?)

Researchers complain that they spend their time filling out research proposals, attending boring meetings in the ill-ventilated rooms of Brussels' Berlaymond building, and writing reports. Some even pretend that the contribution of Brussels is more than offset by the additional administrative work it causes. People are beginning to realize more than ever that money alone is not enough. Clever minds cannot be generated on the spot by cash. Finally, control is weak: Once the project has been approved, Brussels says little about the results.

So how do programs succeed? The winners appear to contain participants that would have collaborated anyhow. The programs just become an opportunity to pocket some additional ECUs. Also, a clear goal with a tangible result, like shooting a space shuttle into orbit before 1995, characterizes successful programs. Finally, successful programs require the members to be personally compatible. Members must be friends, or programs don't work.

All things considered, the Jessi (Joint European Submicron Silicon) program of Eureka seems to be on the right track. The electronics giants of Europe—Philips of the Netherlands, CGS-Thomson of France, and Siemens of Germany, among others—have joined forces to develop the technology for a 64-megabit, 0.3-micrometer RAM before 1996. The European community agreed in June to support this program with 4 billion dollars. Jessi answers Sematech, a semiconductor manufacturing consortium in the US made up of its own giants. Jessi also is Europe's retort to the huge spending of statelike Japanese firms.

In the meantime, seven leading US manufacturers have cofounded US Memories, Inc., as a response to Jessi and to the Japanese domination of the dynamic RAM business.

Table 2.
Summary of Jessi goals.

Specifications	Time periods 1990-1992	1993-1995
Structure width	0.5 micrometers	0.4-0.3 micrometers
Chip area	200 millimeters^2	500 millimeters^2
Memory elements/chip	40,000,000	200,000,000
Logic elements/chip	2,000,000	10,000,000

The success of the Jessi program also rests on the exploitation of a huge silicon market that is growing at 14 percent per year, according to Jessi's bureau. Analysts expect the market to reach 160-billion US dollars by the year 2000, with a 28-billion US-dollar share for Europe. The estimates for the 1992 European market for silicon reach 10 million US dollars. Europe could secure millions of jobs in this area and associated domains.

The Siemens-Philips team has previously shown its combined strength in the Mega-project. Siemens developed a 1-Mbit DRAM, while Philips produced a 256-kilobit static RAM (both in 1.2-micrometer versions.) They used the same technology—a static RAM just takes four times more space than a dynamic one. Siemens began production this year in Regensburg, Germany, becoming (after IBM) the second company to manufacture 1-Mbit DRAMs outside of Japan. IBM, in the meantime, had already started its 4-Mbit DRAM production in Sindelfingen, Germany (luring Chancellor Helmut Kohl into the cleanroom as a special public-relations attraction).

The Jessi project also explicitly responds to European fears that the control of this strategic technology could escape them. Two thirds of all ICs in European products come from either the US or Japan. The US could stop exports to Europe for national security reasons, while Japan could stop for economic reasons. The oil crisis of 1973 could one day be followed by a silicon crisis. Already, agreements between Western countries prevent exporting products that contain advanced embedded ICs of US origin to certain countries.

Technical, financial, and emotional motivation has translated into a strong administrative framework for Jessi. The team worked out its first comprehensive plan at the Frauenhofer Institute for Microstructure Techniques in August 1988. They divided the work into several key areas: technology, equipment and materials, application (design tools), and basic research. The administrative effort has been impressive. A 1,200-page green book defines the project goals, which are summarized in Table 2.

Technology projects focus on mastering submicrometer production techniques. Equipment projects should help overcome Europe's traditional weaknesses in cleanrooms, wafers, lithographic equipment, encapsulation, and test procedures. Application projects induce future application-specific ICs. Memories provide the first step in mastering the technology, but in the long run, custom ICs should become the key market. A unified silicon language should allow clients and silicon foundries to speak the same language, yet preserve user know-how from the prying eyes of the silicon maker. Advanced CAD tools, design verifiers, and silicon compilers prepare this phase. Finally, basic research projects should show the way beyond current physical barriers.

Another element that promotes Jessi's chances of success is that industrial firms share in its leadership (academia is virtually nonpresent). The governing board resides firmly in the hands of the founding firms. In fact, a retired director of Philips heads the board. Founding

The success of the Jessi program rests on the exploitation of a huge silicon market.

members of the governing board also benefit from early review of project proposals. This advantage provides a market-analysis tool that allows frequent exposure to good ideas.

The Jessi project invites other firms and countries outside of the European economic community to join the club—provided they bring their own money along. Non-European partners are welcome under two conditions. They must conduct research, development, and production in Europe, and the members must reciprocate in this collaboration.

While IBM easily fulfills the first condition, it does not meet the second. Europeans have no access to US Memories or Sematech. It is likely, however, that a common ground will result from the negotiations this fall between Sematech and Jessi. IBM's experience in the field of producing 4-Mbit DRAMs could outweigh the "reciprocal" restriction of the Europeans. US Memories may also open its doors to Europeans. Its goals seem similar to Jessi's, an experienced IBM manager leads the firm, and IBM's 4-Mbit DRAM will most likely emerge as its first product.

Sematech faces a similar dilemma, with the Japanese NEC Corporation knocking at the door—represented by its US subsidiary, of course.

Those who are interested in learning more about Jessi may contact the organization by writing the Jessi Planning Group, Margarete-Steiff-Weg, 3D-2210 Itzehoe, Federal Republic of Germany.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 174 **Medium** 175 **High** 176

Letters

Macintosh issues welcomed

To the Editor:

My initial reaction to the August issue of *IEEE Micro* was to return the reader's card to you with my comments appended. Subsequently, I surmised that writing you my comments in this short note would be more appropriate.

First, I would like to register my interest as a reader of two departments, Micro News and Micro Review, as numbers 179 and 185, respectively.

With regard to the item in Micro News, "IEEE Standards to launch hypertext series," I will place it on a local Washington, D.C., Macintosh BBS of which I am a member. I believe that there are interested members who may want to become involved in this effort from a Macintosh viewpoint.

My other comments pertain to the "A new Macintosh environment" item written in Micro Review. A very important matter that the reviewer of the Wingz application did not discuss was the configuration of his Mac Plus computer. If the reviewer had system version 6.0.2 installed in his Mac Plus, coupled with the later market version of the Mac Plus with its small computer system interface (SCSI), I would be very surprised if he would have encountered all the problems that he alluded to in his comments. I have used Wingz routinely since it came on the market and I have nothing but accolades for this application. There are some user-friendly options that could be added to Wingz to make it more efficient; nevertheless, compared to Excel, Wingz is a heads-and-shoulders performer.

I take the liberty of enclosing a review of Wingz performed by Frank Potter (with his permission). Potter reviews Macintosh products for the Washington, D.C., Metro Apple Users Group (Washington Apple Pi). The review I have enclosed was his initial cut at reviewing Wingz and I extracted it from the local Macintosh BBS. *[Limited space keeps us from reprinting this review. Interested readers should extract it from the BBS.—Ed.]*

Keep up the good work. I particularly enjoy reading about important issues pertaining to Apple Macintosh products, such as those involving the IEEE standards for hypertext.

Louis F. West
Fairfax, VA

Richard Mateosian, Micro Review editor, replies:

I have system version 6.0.3 installed, and my Mac Plus has a SCSI interface. I purchased it in January 1987.

Upon reviewing my column, the only even vaguely negative things I said about Wingz on my Mac Plus were: "Its fancy graphics were painfully slow on that machine." "I did not care for the 'look and feel' of Wingz nearly as much as that of Excel."

I'm sure that if Louis West will run the Wingz demo program on a Mac Plus, he will find it just as painfully slow as I did. The special graphics features designed to take advantage of an FPU simply cannot run very fast on an unadorned 68000. As to my personal feelings about the 'look and feel' of Wingz, perhaps I should have kept them to myself. I readily admit that such judgments are subjective.

Micro Law

Richard H. Stern
Law Offices of Richard H. Stern
1300 19th Street NW, Suite 300
Washington, DC 20036

Appropriate and inappropriate legal protection of user interfaces and screen displays

Part 3, Copyright law, the courts, and the Copyright Office

Concerns similar to those discussed in the two preceding issues of this magazine—about how copyrights might hinder software progress more than they encourage it—attend any grant of intellectual property protection. The concerns are legitimate, but on balance they are not determinative. There are also strong reasons for according intellectual property protection to technological advances. Protection provides an appropriate level of incentive to innovate. Only a calculus of potential benefits and harms, showing whether the benefits weigh less or more than the harms, can be determinative.

The patent and copyright laws are authorized by the patent and copyright clause (Article I, Sec. 8, Cl. 8) of the United States Constitution. It gives the US Congress power to promote the progress of science and useful arts by securing to authors and inventors, for limited times, the right to exclude others from exploiting the authors' writings and the inventors' inventions.

The US Supreme Court has held these words to be both a grant of power to Congress and a limitation on congressional power. By negative implication from the grant, Congress may not enact intellectual property laws that will hinder the progress of science and useful arts. Congress may not grant perpetual patents or copyrights, and it may not withdraw from the public things already in the public domain. The patent and copyright laws therefore exist to promote the progress of human knowledge. That is their *raison d'etre*.

The copyright laws apply to computer programs. Until 1980, that they applied

Series Highlights

User interfaces for computer programs are often significant in determining how easy it is to learn to use and how efficiently users can utilize the programs. They are therefore significant factors in the commercial success of computer software. Should the law protect screen displays and other aspects of user interfaces from competitive imitation? If so, what is the best mechanism to do so?

The US Copyright Office recently ruled that screen displays should be protected, but only as an "integrally related" part of the code of the computer programs to which they relate. The Computer Society had urged the Office to protect screen displays separately and independently from the code of the computer programs associated with them. A court then adopted a "legal fiction" that the Copyright Office may allow only one registration for the combination of code and screen displays, but the court would treat the registration as if it were two registrations, one for the code and one for the screen displays.

The main question posed by legal protection of screen displays and related aspects of user interfaces concerns two issues. One is the importance of keeping the teachings of the science of human factors analysis (human-computer interaction) freely available to workers in the field. Just as important is the need to provide incentives to creators and investors in innovative screen display technology.

Some court decisions and some aggressive litigation positions of software publishers have raised legitimate industry concerns. People wonder whether copyright protections for screen displays will be used to lock up utilitarian features of screen design, thereby impoverishing the toolbox of other workers in the screen design field.

The concerns are most acute in the case of very important or necessary techniques, such as those dictated by human factors analysis. But concerns may also be appropriate about screen design conventions and emerging conventions. Moreover, these concerns extend beyond screen display expedients to keystroke conventions such as </> (slash) for invocation of command mode.

A very troubling question is whether user habituation to particular user interface or screen display expedients—resulting from users' investments in self-education—should give rise to a legally protected interest.

Part 3 of this series addresses how the courts and Copyright Office are dealing with concerns of this type, after providing a brief copyright law tutorial.

0272-1732/89/1000-0008$01.00 © 1989 IEEE

at all to computer programs was questionable. Congress then passed an amendment to the Copyright Act. In effect Congress extended its protections to "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."

Initially, the courts disagreed over whether these words covered object code as well as source code and whether they covered "ROMmed" code (computer program code stored in a semiconductor chip). But eventually the courts, in effect, interpreted the statutory terms *used directly* to mean machine language and *used indirectly* to mean code that had to be interpreted, compiled, or assembled into machine language to be used in a computer. Moreover, "ROMmed" code was held to be protected under the copyright law.

The patent laws also apply to computer programs, at least when the program is associated with a machine system that does more than just compute numbers. As a practical matter, however, the requirement of technical merit ("nonobviousness"), the $5,000 to $15,000 cost, and the three-year time delay involved in issuance of an electronic-equipment patent have so far tended to make the patent system of little importance for screen displays. (In the last several years, interest has grown in design patents on aspects of screen displays, such as icons. Design patents protect ornamental, nonfunctional aspects of products. They usually issue in less than a year and cost approximately $1,000. The validity and scope of design patents on screen displays are yet to be tested in the courts.)

Because patent protection on screen displays is as yet untested and uncertain, this series focuses on copyright aspects of intellectual property protection of screen displays, to the exclusion of other forms of intellectual property.

Copyrights. The copyright laws seek to promote human knowledge by providing an economic incentive to those who create new works of authorship or who subsidize such creation by their investment in creation. (Subsidies include paying the salaries of programmers and designers of user interfaces for software.) The owner of a copyright receives the legal right to exclude all unauthorized persons from reproducing and marketing the copyrighted work.

In theory, the right to exclude applies only against those who copy the work, as contrasted with those who independently create the work without ever having had access to the protected work. The distinction is illusory, however, in the case of the screen displays of a widely marketed computer program. In this case everyone in the trade will know about the computer program and thus will have had access to its screen displays. Independent creation is at best a theoretical possibility.

The owner's exclusive right is called a monopoly if you are suspicious of it, or a proprietary right or property right if you are enthusiastic about it. However you characterize the right, it works by giving its owner the legal power to exact "economic rent" from the exploitation of that right.

Paying some economic rent to creators and developers of new technology will mobilize inventors to their attics, engineers to their garages, and investors to their wallets.

Consider for example, disks containing the Lotus 1-2-3 spreadsheet program or the Ashton-Tate dBase III database management program, and the accompanying documentation. Such software packages might sell for barely more than the cost of the blank diskettes and the cost of photocopying the manuals (a few dollars instead of a few hundred dollars), were it not for the copyright laws. The difference between the going market price and those costs represents the copyright owners' economic rent. The right to this rent compensates them for creating the product and putting it on the market, and at the same time provides an incentive to others to become similar entrepreneurs.

This incentive provides the underlying theory or premise of intellectual property law, which applies to patent law and copyright law alike. The theory is that paying some economic rent to creators and developers of new technology, in the form of a grant of an exclusive right, helps everyone. It will mobilize inventors to their attics, engineers to their garages, and investors to their wallets to underwrite creative costs. Eventually, the public benefits because the result will be a greater amount of innovation than would occur in the absence of the system of exclusive rights.

It is implicit in the theory that the social cost will be less than the social value of the incremental amount of creation caused by maintaining the system. The social cost of this system is the sum total of economic rent paid to creators and their investors, in the form of copyright royalties, higher prices for copyrighted products, lost value from hindering advances by others, or whatever. While there is little hard empirical evidence one way or the other on this point, customary practice by Congress and the courts has made this premise a fixture of intellectual property law. It is beyond the scope of this series to subject that premise to further or more rigorous analysis.

Enforcement of copyrights. A US copyright owner may enforce its copyright only after registering the work with the US Copyright Office. Registration involves submitting a completed copyright application form to the Copyright Office, providing it with a copy of the work or of material identifying it, and paying a small fee (ordinarily $10). Within a few months the Office will examine the material for copyrightability, in a limited way without the kind of inquiry given a patent application. It will then issue a certificate of registration of a claim of copyright. The copyright owner may then sue any copyright infringers in federal court. An infringer is a person who reproduces or distributes the work without authorization from the copyright owner—or, in copyright owners' preferred parlance, a pirate.

If a judgment of copyright infringement is entered against a pirate, the copyright owner may seek an injunction against continuation of the infringement, damages for lost sales, and an award of the pirate's ill-gained profits. The copyright owner may also secure an order for the destruction of infringing goods and for destruction of equipment used to commit the copyright infringement, attorney fees, and other relief.

An intentional copyright infringement, committed for purposes of commercial gain, may subject the infringer

continued on p. 75

SPECIAL FEATURE

An Efficient Scheme for Interprocessor Communication Using Dual-Ported RAMs

Need a simple, fast, modular system to transfer messages in a multiprocessor? Try this low-cost duplex scheme.

N. Jagadish
J. Mohan Kumar
L.M. Patnaik

Indian Institute of Science

Successful interprocessor communication, a key factor in the design of any multiprocessing system, requires high bandwidth and reliability with minimal cost and software/hardware overheads. Here, we present such a communication scheme. It features simplicity, speed, modularity, and configurability to multiprocessing systems such as linear arrays, triangular arrays, meshes, systolic trees, and hypercubes.

Communication between any two processors in this scheme takes place through a common memory, independently accessible by both processors involved. The interprocessor interconnection scheme in a multiprocessor system directly affects system throughput and has a bearing on the modularity, reliability, and overall system performance. Yalamanchili and Aggarwal discussed the importance of the processor interconnection scheme when they characterized the capabilities of a multiprocessing system.[1]

Various interconnection schemes have been suggested for message passing between processor nodes.[2,3] Tuazon et al.[2] suggested a scheme that makes use of first-in, first-out, or FIFO, buffers and several communication channels. Their scheme involves data-shifting mechanisms and software for polling signals. In this scheme the transfer of a message between two processor nodes involves 1) transferring a message to the FIFO buffers in the source node, 2) converting a message from words to nibbles, 3) transmitting a message from source node to destination node, 4) reconverting the message from nibbles to words in the destination node, and 5) receiving message data from the FIFO buffer in the destination node after checking data-valid flags.

In the Hayes et al. scheme,[3] processor nodes communicate with one another by means of asynchronous direct memory access operations. The message moves through serial channels. Transmission involves 1) DMA transfer from the main memory to a buffer on the processor node, 2) conversion of the message into serial format, 3) transmission on a serial communication channel, 4) reconversion of the message into parallel format, and 5) another DMA transfer from a buffer to the main memory on the destination node.

0272-1732/89/1000-0010$01.00 © 1989 IEEE

The Intel iPSC,[4] a hypercube supercomputer, consists of eight communication channels per node. Intel built the Ethernet protocol-based iPSC using a special local communication coprocessor (82586). The iPSC scheme transfers data at the rate of 10 Mbits per second. Carrier Sense Multiple Access with Collision Detection (CSMA/CD), a statistical medium access control system, implements the sharing of common channels.

Our scheme for interprocessor interconnection using dual-ported RAMs and network controllers follows. In this scheme, communication between the processor nodes involves writing into and reading from a common memory area. The communicating processors do not have to contend for a common bus as in the case of shared-memory systems, since they have independent access to the common memory units shared between them. Only the memory access time of the processors limits the communication speed. Processor-to-processor communication does not use intermediate buffers, input/output ports, or DMAs. We consider the example of a three-dimensional cube to illustrate the advantages of this scheme. Further, we discuss the implementation of the interprocessor communication scheme on a 64-node cube configuration.

Processor-to-processor communication

Dual-ported RAMs now available in VLSI form operate at static RAM speeds (50 to 150 nanoseconds) and have two independent left and right ports. Figure 1 illustrates a message transfer between two neighboring processors using dual-ported RAMs. This DPR area is common to both processor elements (PE1 and PE2). In other words a portion of the memory space of PE1 "overlaps" a portion of the memory space of PE2. We shall refer to this shared DPR area as the common memory.

Either processor can access the dual-ported RAM independently, since this memory area lies in the memory space of each processor. However, PE1 and PE2 access this area with different addresses. On-chip arbitration logic within the dual-ported RAM handles address contention to ensure maximum speed. In case of contention one of the ports must wait until the other port's access is complete; a BUSY signal on the dual-ported RAM indicates contention.

As shown in Figure 1, the common memory between PE1 and PE2 is logically divided into upper and lower halves. PE1 writes into the upper half and reads from the lower half. Similarly, PE2 writes into the lower half and reads from the upper half. In this way we minimize the probability of access contention. To transmit a message packet to PE2, PE1 writes the message packet in the common memory it shares with PE2. Communication between neighboring processors does not involve intermediate devices. A message written by the transmitting processor in its own memory is accessible to the receiving processor.

Communication between noncontiguous nodes (those not directly connected with each other) can be carried out with the help of an intermediate processor.[2] Figure 2 illustrates the methodology for communication between noncontiguous processors. A network controller (NC) becomes the intermediate link for message transfer between noncontiguous nodes. (We

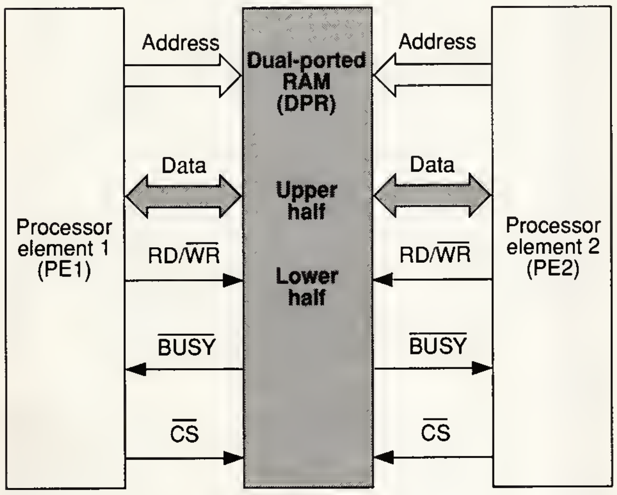


Figure 1. Two processors sharing a dual-ported RAM.

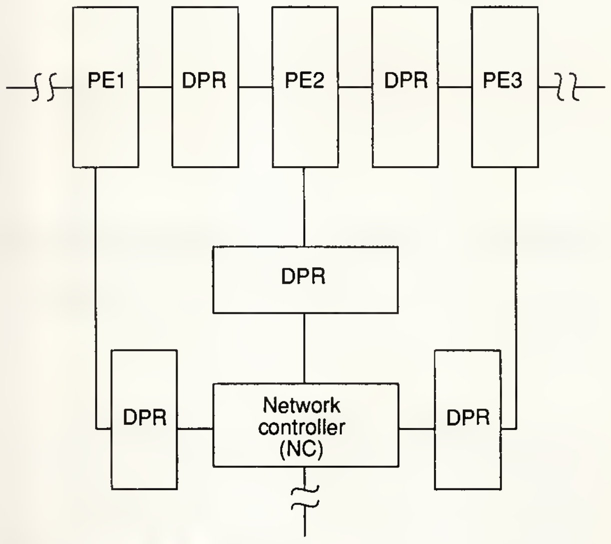


Figure 2. Communication pathways among noncontiguous processors.

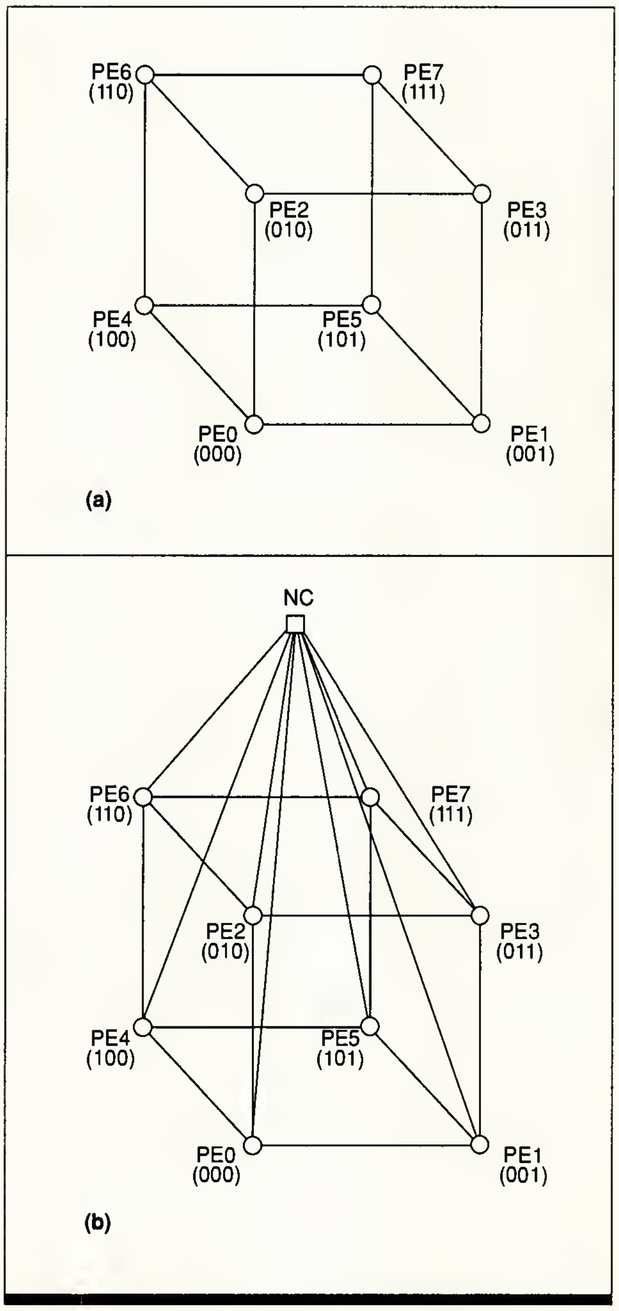


Figure 3. A 3D cube (a) and a 3D cube with a network controller (b).

show the usefulness of the network controller in hypercube configurations in Figure 3 and discuss it later.)

In Figure 2, two noncontiguous processors PE1 and PE3 share separate dual-ported RAMs with the network controller. This controller provides an alternative path for the transfer of messages between two noncontiguous nodes. As indicated in Figure 2, PE1 can transfer messages to PE3 along two alternative paths: one path via PE2, which involves two memory transfers through PE2, and another path through the network controller. In the first method, PE2 participates in the message transfer between PE1 and PE3.

In a multiprocessing system dedicated PEs perform subtasks of a main task. The throughput of the multiprocessing system would be significantly reduced if the PEs were used for communication purposes. We include the network controller seen in Figure 2 to transfer messages between noncontiguous nodes. To transmit a message packet from PE1 to PE3, PE1 writes the message packet into the common memory shared with the network controller; this controller performs a block transfer to shift the message packet to the common memory shared with PE3.

With this kind of design, the PEs in a multiprocessing system need not participate in the communication between noncontiguous nodes as the network controller exclusively performs communication tasks. Obviously, processor-to-processor transfer is most effective in the case of contiguous nodes. Message transfers between noncontiguous nodes must use the path through the network controller. Thus the two paths for message transfers complement each other.

Implementation on a 3D cube

To understand the implementation aspects of our scheme, we suggest a three-dimensional cube, since such a topology has attracted wide interest among researchers in recent years. An n-dimensional hypercube[5] is a multiprocessor characterized by the presence of $N = 2^n$ processors interconnected as an n-dimensional binary cube. Each node of the cube consists of a central processing unit and local main memory. Each PE of the cube directly communicates to n other PEs of the cube; the communication paths correspond to the edges of the cube. The length of the path between any two nodes is simply the number of edges of the path. The minimum distance between any two nodes in an n-cube equals the Hamming distance between them.

Implementing the interconnection network on a 3D cube occurs as follows. The nodes of the cube are numbered as indicated in Figure 3a and b. Each node consists of a PE, which includes the numeric data coprocessor. Each PE shares common memory units with other PEs located at a Hamming distance of one. In addition, the PEs share common memory units with the network controller, as illustrated in Figure 4a. Figure 4b shows the memory map of a typical node processor.

As can be seen in Figure 4c, the processing node contains an 8088 processor with an 8087 numeric data coprocessor, address decoding logic, a wait-state generator, system ROM, local RAM, and dual-ported RAMs. The address decoding logic selects among the system ROM, local RAM, and the dual-ported RAMs

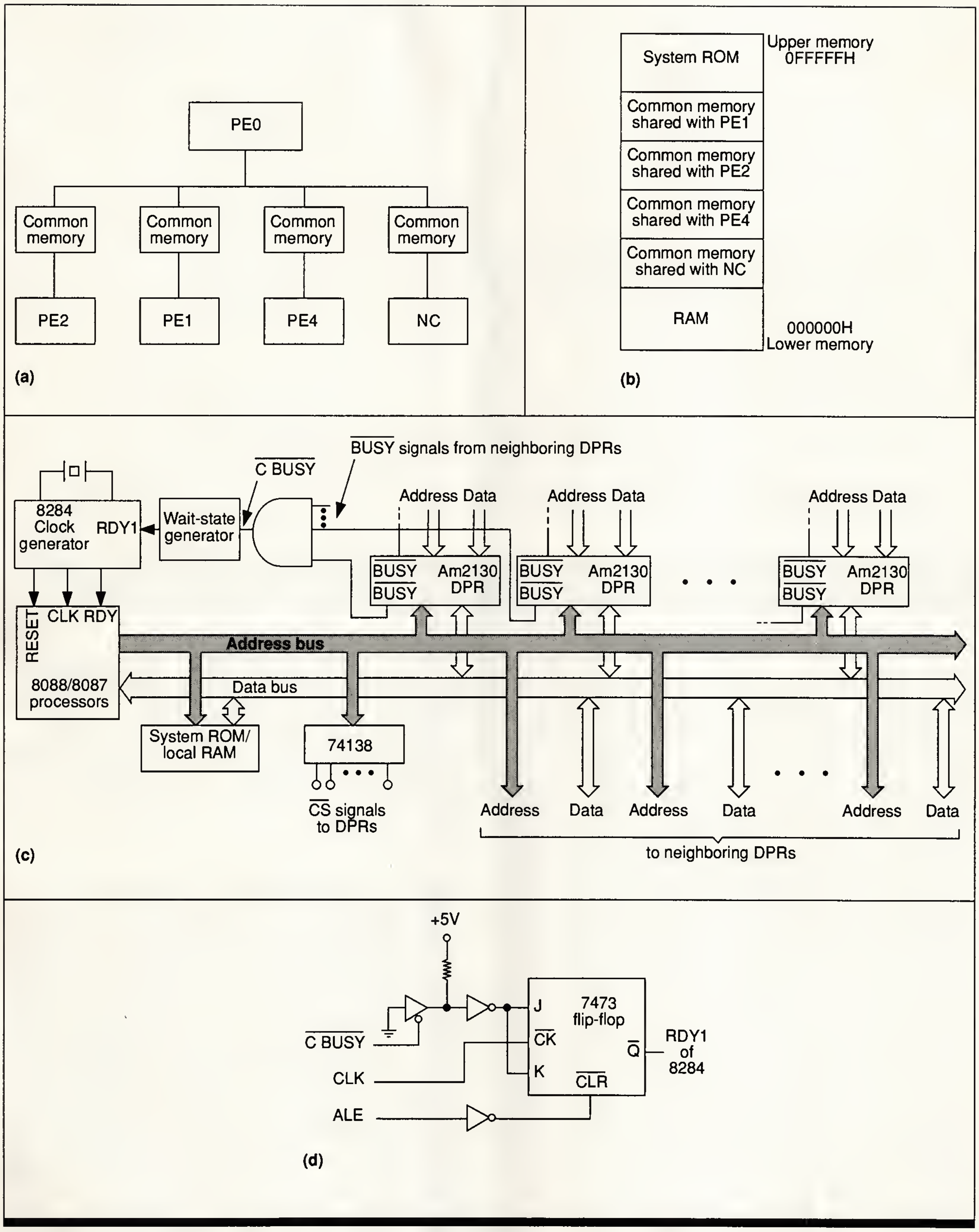


Figure 4. Common memory units at a typical node called PE0 (a); a memory map of PE0 (b); node processor hardware (c); and a single-wait-state generator (d).

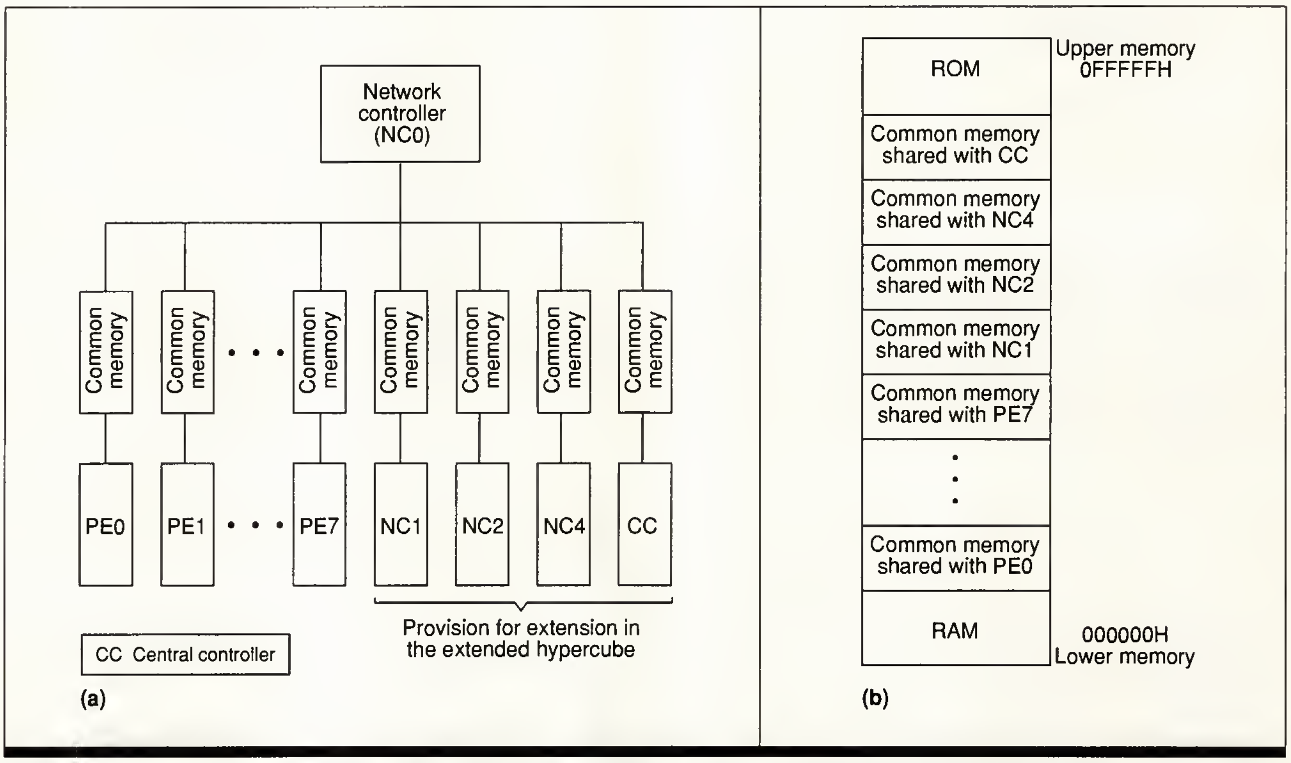

Network controller (NC0)
Common memory
PE0
PE1
PE7
NC1
NC2
NC4
CC
Provision for extension in the extended hypercube
CC Central controller
(a)
ROM
Upper memory 0FFFFFFH
Common memory shared with CC
Common memory shared with NC4
Common memory shared with NC2
Common memory shared with NC1
Common memory shared with PE7
Common memory shared with PE0
RAM
000000H Lower memory
(b)

Figure 5. Common memory units at a network controller (a) and a memory map of the network controller (b).

shared between the neighboring processors and the network controller. The wait-state generator takes care of address contention as follows. On detection of a BUSY signal from the dual-ported RAM, the wait-state generator disables the ready-line input RDY1 of the 8284 clock generator for one clock state. See Figure 4d.

The network controller is dedicated to the task of overall interprocessor communication management. The controller shares common memory units with each of the eight nodes in the cube, as illustrated in Figure 5a and b, and its hardware configuration is similar to that of the nodes. As indicated in Figure 6, the network controller contains parallel and serial ports for communication with the host system and other input and output devices. In addition, the network controller initializes the cube and distributes tasks.

Common memory units exist between all pairs of neighboring nodes and between the network controller and each node. As mentioned earlier, neighboring nodes communicate by directly writing into the common memory located between the two nodes. For communication between nodes located at a Hamming distance greater than one, the network controller performs a memory block transfer from the common memory shared with the transmitting node to the common memory shared with the receiving node. A message packet between two noncontiguous nodes can also be routed through one of the parallel paths between the two nodes depending on the availability of the processors.[5] The parallel paths between two noncontiguous nodes may consist of one or more nodes that contribute to the message transfer by block transferring the message packet from the memory space of the transmitting node to the memory space of the receiving node.

Message transfer protocol

The message packet shown in Figure 7a consists of the semaphore/address byte, packet-size byte, and the actual message. The semaphore/address byte (Figure 7b) has three subdivisions. The most significant bit indicates valid data, the next bit indicates processor-busy status, accompanied by three bits for addressing processors in an extended hypercube, and the last three bits indicate the address of the node. The next byte gives the total length of the message in bytes, followed by the message itself. The source node checks the semaphore bits for data validity and writes the message packet either in its common memory shared with the destination processor (if the Hamming distance between them is one), or in its common memory shared with the network controller (if the Hamming distance between the source and destination nodes is greater than one).

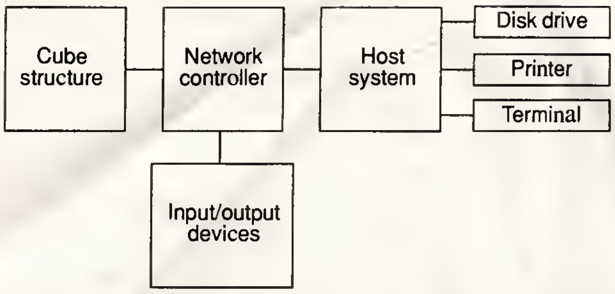


Figure 6. Block diagram of the multiprocessor system with peripherals.

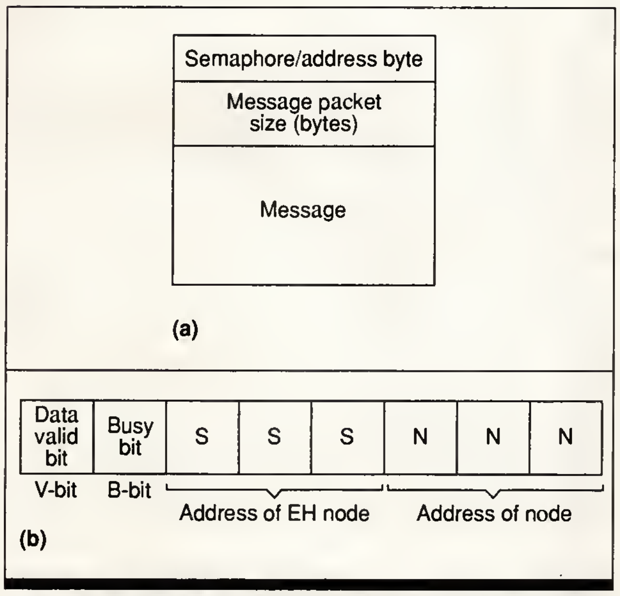


Figure 7. A message packet (a) and a semaphore/address byte (b).

```
Procedure Initialization for network controller
    for PE0 to PE7
        begin
            V-bit := 0; B-bit := 0;
        end;

Procedure Network Controller Block Transfer
  repeat
    for PE0 to PE7
        begin
            if ((S-V-bit) = 1) AND ((D-V-bit) = 0) then
                block transfer data;
                S-V-bit := 0; D-V-bit := 1;
        end;
  forever.

Procedure Send  /* for PE */
        begin
            if hamming distance >1 then
                block transfer data to network controller;
                network controller-V-bit := 1;
            else
                block transfer data to destination;
                D-V-bit := 1;
        end;

Procedure Receive  /* for PE */
        begin
            if S-V-bit = 1
            then
                block transfer data;
                S-V-bit := 0;
        end;

/*  B-bit is busy bit
    S-V-bit is source data validity bit
    D-V-bit is destination data validity bit
    Network controller V-bit is network controller data
    validity bit  */
```

Figure 8. The procedure for a message transfer.

As indicated earlier in Figure 3b, PE3 and PE7 (with a Hamming distance of one between them) share a common memory through which they can communicate. The source processor checks the data valid V-bit. If the V-bit is 0, the source processor writes the message in the common memory space and sets the V-bit to 1 to indicate the presence of fresh data. The destination processor checks the V-bits in the common memory units shared with its neighbors. If any of the V-bits are valid, the processor copies into the local memory the message from the common memory following a valid V-bit, and resets the V-bit to 0.

The message transfer between two noncontiguous nodes involves a path through the memory space of the network controller. For example, consider the communication procedure between PE3 and PE4, which are at a Hamming distance of three from each other. The source processor PE4 determines the Hamming distance between itself and the destination processor PE3. Since the Hamming distance is greater than one, PE4 writes the message into the common memory space shared with the network controller if the corresponding V-bit is reset to 0. The network controller checks the V-bits of the control bytes in the common memory units shared between PE4 and PE3. Say the V-bit of PE4 is 1 (indicating fresh data) and the V-bit of PE3 is 0 (indicating previous data accepted). In this case, the network controller transfers a memory block to shift the data stored in the common memory space shared with PE4 to the common memory space shared with PE3. The procedure for message transfer appears in Figure 8.

Scheme extended to 64 nodes

The 64-node extended hypercube, or EH, consists of eight 3D cubes[5] and a central controller node, as illustrated in Figure 9. (We introduce the EH term to reflect that each node of the hypercube is a cube by itself.) Each 3D cube consists of eight individual nodes and the network controller, and we refer to this group as the EH-node (node of the EH). EH-nodes appear at the vertices of the EH. Each of the eight EH-nodes has topological and architectural features similar to that of the 3D cube discussed earlier.

As indicated by the dotted lines in Figure 9, the eight network controllers at the eight EH-nodes form a 3D cube, the EH. The network controller of each EH-node shares common memory units with its neighboring network controllers in the EH. In addition, a central network controller shares common memory units with all the eight network controllers at the vertices of the EH. There is no interconnection network between the individual nodes of different EH-nodes.

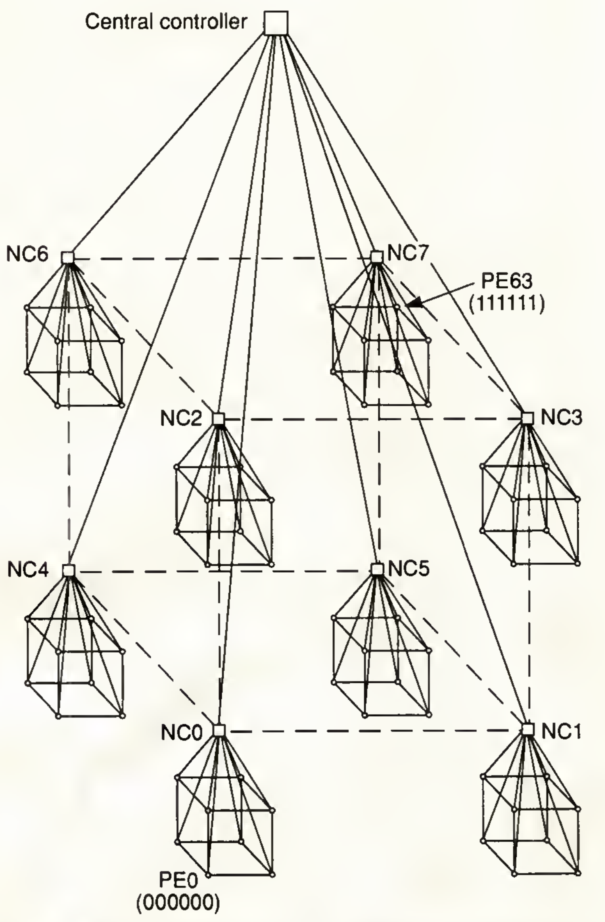


Figure 9. An extended hypercube.

The interprocessor communication scheme between individual nodes of the EH-node is similar to that explained earlier, as is the communication scheme between any two network controllers in the EH consisting of the network controllers and central controller.

A message between two individual nodes in different EH-nodes transmits via the memory space of the network controllers residing in the EH-nodes of the source and destination nodes. A message transfer between two individual nodes having a Hamming distance of six between them (and residing in two different EH-nodes that are themselves at a Hamming distance of three from each other) completes with just three memory transfer operations. No individual node processor (other than the source and destination node processors) participates in the memory transfer operation. Memory transfers can best be explained by considering two such nodes shown in Figure 9, PE0 and PE63, as source and destination nodes respectively. Messages transfer as follows:

1) PE0 writes the message packet in its common memory shared with NC0,
2) NC0 transfers the message packet to the common memory shared with the central controller,
3) the central controller performs another memory block transfer to shift the message packet to the common memory shared with NC7, and
4) NC7 performs a final memory transfer to place the message packet in the memory space of the destination node.

The SSS bits in the semaphore/address byte as shown in Figure 7b indicate the address of the EH-node. The network controllers in each EH-node keep track of the busy status of the individual PEs, and the central controller keeps track of the busy status of the network controllers.

Advantages of the scheme

The dual-ported RAM scheme presents a cost-effective method for data transfer between processor nodes in a multiprocessing system. Tuazon et al. discussed a scheme that yields a data transfer rate of 1.5 Mbits/second. Hayes et al. discussed another scheme using DMAs and serial channels with a transfer rate of 1 Mbyte/s. During DMA transfers, though, the processor must remain idle until the DMA transfers complete. Software overhead may further reduce the effective data transfer rate. The CSMA/CD scheme employed by the iPSC cube offers a data transfer rate of 10 Mbits/s, but has overhead related to the special communication coprocessor and its related initialization and control software.

In our scheme, any pair of processors of a hypercube can establish two-way communication. A processor can receive messages from three of its neighbors and

Table 1.
Data transfer rates for different processors.

Processor	Clock (MHz)	Key instruction	Overheads (incl. states)	Typical transfer rates (Mbytes/s)
8088	10	REP MOVS	33	0.58
8086	10	REP MOVS	33	1.17
80286*	10	REP MOVS	19	5.00
80386*	16	REP MOVS	21	16.00
68000	12.5	MOVE.l (a1)+,(a2)+	50	1.78

*In real-address mode

Table 2.
Instructions for data transfer.

Instruction	Clock states required
MOV AX, DATA_SEG	4
MOV DS, AX	2
MOV AX, EXTRA_SEG	4
MOV ES, AX	2
MOV CX, LENGTH_OF_PACKET	4
MOV SI, SOURCE_POINTER	4
MOV DI, DESTN_POINTER	4
REP MOVS	9 + 17(*n*)
Total no. of clock states	33 + 17(*n*)

n is the number of byte transfers.

the network controller and send a message to one of its three neighbors or the network controller simultaneously. In other words, at a given time four communication paths of a PE can be active. One of these four can be a two-way communication path. In an 8-node cube with 20 memory units, nine paths can be active at any given time. For example, in Figure 3b PE5 can receive messages from three of its neighbors (PE1, PE4, PE7) and the network controller. PE5 can send a message to one of its neighbors, while other PEs (PE0, PE2, PE3, PE6) can have four active communication paths among them.

We implemented the dual-ported RAM scheme with Intel 8088s as node processors because of the availability of hardware/software development tools and the hardware's low cost. The high-speed communication technique has advantages in a multiprocessing system. The technique can be adopted for communication in multiprocessing systems based on advanced microprocessors like Intel's iAPX 80286 and 80386 and Motorola's 68000, 68020, and 68030. Table 1 lists expected zero-wait-state data transfer rates when using typical instructions. This transfer rate is dependent on the bus bandwidth and the type of instructions available. With an 8088 processor operating at 10 MHz, we obtained a zero-wait-state transfer rate of 0.588 Mbytes/s (4.7 Mbits/s) in both directions (duplex). This transfer rate from one PE to another is computed as shown in Table 2. The REP MOVS string operation essentially achieves a block move of data from one part of memory to another.

The software overhead for initializing the various registers involves 33 clock states. The transfer rate actually depends on the REP MOVS instruction, which takes 17 clock states per transfer of a byte in the case of the 8088. This, when computed for a processor operating at 10 MHz, yields 0.588 Mbytes/s or 4.7 Mbits/s. The speed improves significantly if processors with wider data bus widths and higher clock frequencies are used.

Our fully duplexed, asynchronous, and zero-buffered communication scheme handles messages that are less than the maximum allowable packet size. Processor nodes operating at different speeds and different word lengths could be combined in the same multiprocessor system. The highly optimized dual-port technique allows the same memory to be used as working storage and for communication between nodes, avoiding the need for any special data communication controller. Message transfer is transparent to the user programs running on the nodes because no special communication channel must be set up and no need exists to keep track of packet sequence. We further reduce software overhead in that we do not need acknowledgment packets for memory-to-memory transfer. Advanced processors with higher addressing capability can support communication channels with larger sizes of dual-ported common memory and hence improve the throughput.

In an n-cube configuration each individual node connects to n neighboring nodes.[5] A message transfer operation between any two nodes with a Hamming distance of n involves $(n - 1)$ processor elements and transmission on n links. We discussed the implementation aspects of our scheme on an eight-node (2^3) cube and the extension of the scheme to a 64-node cube (2^6 cube) configuration. With the help of the dual-port technique and the use of the network controllers, a message can transfer between any two nodes in a 2^3 cube with a maximum of two memory transfers, even if the Hamming distance between the nodes is three. In the 64-node architecture, we achieve a message transfer between any two communicating nodes with a maximum of three memory transfer operations. A message can also be transmitted from a source node to a destination node through one of the several parallel paths consisting of the PEs, network controllers, and the central controller, depending on their availability.

The dual-ported RAM approach for message transfer between nodes in a multiprocessor system offers cost and speed advantages. The extended hypercube is an example of a low-cost, compact multiprocessor system with minimal software and hardware overheads. With an 8088 processor operating at 10 MHz, we have achieved a data transfer rate of 0.588 Mbytes per second (4.7 Mbits per second).

Acknowledgments

We wish to thank R. Govindarajan for valuable discussions concerning our work. The suggestions provided by the reviewers and the editor have helped us improve the article's organization.

References

1. S. Yalamanchili and J.K. Aggarwal, "A Characterization and Analysis of Parallel Processor Interconnection Networks," *IEEE Trans. Computers*, Vol. C-36, No. 6, June 1987, pp. 680-691.
2. J. Tuazon, J. Peterson, M. Prifet, and D. Liberman, "Caltech/JPL Mark II Hypercube Concurrent Processor," *Proc. Int'l Conf. Parallel Processing*, IEEE CS Press (microfiche), Los Alamitos, Calif., Aug. 1985, pp. 666-678.
3. J.P. Hayes et al. "A Microprocessor-based Hypercube Supercomputer," *IEEE Micro*, Vol. 6, No. 5, Oct. 1986, pp. 6-17.
4. E.I. Organick, "Algorithms, Concurrent Processor, and Computer Science Education," *ACM SIGse Bulletin*, ACM, New York, Vol. 17, No. 1, Mar. 1985.
5. Y. Saad and M.H. Schultz, "Topological Properties of Hypercubes," Research Report, Dept. of Computer Science, Yale University, New Haven, Conn., June 1985.

Additional reading

Hayes, J.P., et al., "Hypercube Computer Research at the University of Michigan," *Proc. Second Conf. Hypercube Multiprocessors*, Univ. of Michigan, Ann Arbor, Sept.-Oct. 1986.

iAPX 286 User's Manual, Intel Corp., Santa Clara, Calif., 1984.

iAPX 80386 Hardware Reference Manual, Intel Corp., 1987.

iAPX 86/88 User's Manual, Intel Corp., 1983.

Osborne, A., and G. Kane, *Osborne 16-Bit Microprocessor Handbook*, McGraw-Hill, Berkeley, Calif., 1981.

Peterson, J.C., et al., "Mark III Hypercube-Ensemble Concurrent Computer," *Proc. Int'l Conf. Parallel Processing*, IEEE CS Press (microfiche), 1985, pp. 71-73.

Seitz, C.L., "The Cosmic Cube," *Comm. ACM*, Vol. 28, No. 1, Jan. 1985, pp. 22-33.

N. Jagadish is a scientific assistant in the Department of Computer Science and Automation at the Indian Institute of Science in Bangalore. Previously, he was involved in building a part of the processor-based controls for motion of a 90-inch optical telescope at the Indian Institute of Astrophysics, also in Bangalore. His areas of interest include local area networks, performance evaluation, and multiprocessor systems.

Jagadish received the BE degree in electronics and communication from the Mysore University in Mysore, India. Currently, he is working toward the MSc (engineering) degree in computer science at the Indian Institute of Science.

J. Mohan Kumar is a scientific officer in the Microprocessor Applications Laboratory at the Indian Institute of Science and has been a lecturer at Bangalore University. His research interests include multiprocessor systems, neural networks, and parallel computing.

Mohan received the BE degree in electrical engineering from Bangalore University and the MTech degree from Indian Institute of Science, where he is working toward his PhD degree in computer science.

L.M. Patnaik, a professor of the Department of Computer Science and Automation, also chairs the Microprocessor Applications Laboratory at the Indian Institute of Science. His teaching, research, and development interests have been in the areas of parallel and distributed computing, computer architecture, computer graphics, computer-aided design of VLSI systems, and expert systems.

Patnaik obtained his PhD in the area of real-time software for industrial automation. He also holds the DSc degree for his research work in computer systems and architectures. He has published over 150 publications in refereed international journals and conference proceedings and coauthored a book on functional programming. He is a senior member of the IEEE and the Computer Society of India, a founder member of the executive committee of the Association for Advancement of Fault-Tolerant and Autonomous Systems, and a fellow of the Indian Academy of Sciences, National Academy of Sciences, and the Institution of Electronics and Telecommunications Engineers in India.

Questions concerning this article can be directed to L.M. Patnaik, Indian Institute of Science, Department of Computer Science and Automation, Bangalore 560 012, India.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 150 **Medium** 151 **High** 152

SPECIAL FEATURE

Implementation and Performance of the Fast Hartley Transform

You can cut computation time by substituting this easy DSP tool in general applications that involve real signals.

Tho Le-Ngoc
Minh Tue Vo
Condordia University

The Fourier transform, an important tool in digital signal processing, allows signals to be treated in the complex frequency domain. In most applications, however, the signals in the time domain are real, causing the Fourier transform to contain redundancy.[1]

In 1942 R.V.L. Hartley[2] formulated a real integral transform, which later on R.N. Bracewell,[3] O. Buneman,[4] and others extensively investigated. The Hartley transform, because it works with real instead of complex numbers, reduces computation time and thus becomes a potential substitute for the Fourier transform in general applications.

With this in mind, we investigate the implementation aspects of the fast Hartley transform, in both software and hardware. We describe the modifications required to convert existing fast Fourier transform (FFT) programs to execute fast Hartley transforms (FHT), showing the ease with which these modifications can be implemented. We compare execution time and memory storage requirements of both transforms and present power spectrum calculation and convolution as illustrative examples to compare the performance of the two transform techniques. We also survey the comparative performance of various microprocessors and digital signal processors in FFT and FHT computation.

Overview

Bracewell defines the Fourier transform of a signal $f(t)$ as[5]

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}\,dt \\ &= \int_{-\infty}^{+\infty} f(t)(\cos \omega t - j \sin \omega t)\,dt \end{aligned} \qquad (1)$$

where t is the time-domain variable, and ω is the frequency-domain variable. The inverse Fourier transform is given by

$$f(t) = 1/(2\pi) \int_{-\infty}^{+\infty} F(\omega)e^{+j\omega t}\,d\omega$$

0272-1732/89/1000-0020$01.00 © 1989 IEEE

When the input signal $f(t)$ is real, the Fourier transform will be Hermitian;[1] that is, $F(-\omega) = F^*(\omega)$, where the superscript * denotes complex conjugation. Therefore, since half the transform is redundant, the transform time can be reduced.

To exploit this symmetry and redundancy in the Fourier transform of real signals, Hartley[2] introduced the following transform:

$$\begin{aligned} H(\omega) &= \int_{-\infty}^{+\infty} f(t) \text{ cas } \omega t \, dt \\ &= \int_{-\infty}^{+\infty} f(t)(\cos \omega t + \sin \omega t) \, dt \end{aligned} \tag{2}$$

This transform is in some way a real version of the Fourier transform, with the complex exponential function replaced by the cas function ($\text{cas } x = \cos x + \sin x$). Bracewell[1,3] gives us the inverse Hartley transform as

$$f(t) = 1/(2\pi) \int_{-\infty}^{+\infty} H(\omega)(\cos \omega t + \sin \omega t) \, d\omega$$

This is the same form as the forward Hartley transform in Equation 2, except for a scaling factor.

The simple identity $j = -\{(1-j)/(1+j)\}$ yields

$$\begin{aligned} e^{-j\omega t} &= \cos \omega t - j \sin \omega t \\ &= \{(1+j) \cos \omega t + (1-j) \sin \omega t\}/(1+j) \end{aligned} \tag{3}$$

Using this relationship and Equations 1 and 2, we can show that

$$F(\omega) = \{H(\omega) + jH(-\omega)\}/(1+j)$$

which we can also rewrite to yield

$$\begin{aligned} F(\omega) &= [\{H(\omega) + H(-\omega)\}/2] \\ &\quad - j[\{H(\omega) + H(-\omega)\}/2] \\ &= H_E(\omega) - jH_O(\omega) \end{aligned} \tag{4}$$

Hence, $Re\{F(\omega)\} = H_E(\omega)$, the even part of $H(\omega)$, and $Im\ \{F(\omega)\} = -H_O(\omega)$, the negative odd part of $H(\omega)$. Since the Fourier transform can be recovered from the Hartley transform, no information is lost if we use the Hartley transform for a real input signal instead of the Fourier transform.

The discrete transforms. In digital signal processing we convert the signal into a sequence of discrete samples. If the sampling period is T, a time signal $x(t)$ in discrete form becomes $x(nT)$, where n is an integer. (Usually we drop the sampling period T if there is no confusion.) We can therefore redefine the Fourier and Hartley transforms in discrete form.

We achieve the discrete Fourier transform (DFT) of a sampled signal $x(n)$, $n = 0, \ldots, N-1$, by[6]

$$F(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{5}$$

for $k = 0, \ldots, N-1$, and the inverse transform is

$$x(n) = (1/N) \sum_{k=0}^{N-1} F(k) e^{j2\pi kn/N}$$

If the signal $x(n)$ is real, its DFT is Hermitian; that is, $F(N-k) = F^*(k)$ for $k = 0, \ldots, (N/2) - 1$. We can exploit this property to halve the DFT computation time. However, since the DFT of a real signal is generally complex, and since this complex DFT is the input to the inverse DFT, the inverse transformation time cannot be reduced by the Hermitian property, even for a real signal.

We can define the discrete Hartley transform (DHT) of a real signal $x(n)$ as follows:[1]

$$H(k) = \sum_{n=0}^{N-1} x(n) \text{ cas } (2\pi kn)/N \tag{6}$$

for $k = 0, \ldots, N-1$. We have replaced complex multiplications in the DFT formula with real multiplications in the DHT. We reduce the computation time just as if we had used the Hermitian property of the DFT.

According to Bracewell,[1,3] we achieve the inverse DHT by

$$x(n) = (1/N) \sum_{k=0}^{N-1} H(k) \text{ cas } (2\pi kn/N) \tag{7}$$

Note that Equations 6 and 7 have exactly the same form, except for a scaling factor of $1/N$. As a consequence, we can use any computation unit (software program or hardware device) capable of producing the DHT of a signal to calculate the inverse DHT as well. (We do this by including a multiplication by $1/N$). The fact that we have now reduced the computation time (compared to the complex DFT) in both the forward and the inverse transformations further accentuates this advantage of the DHT over the DFT. Equations 3 and 4 still hold if the continuous variable ω is replaced by the discrete variable k:

$$F(k) = \{H(k) + jH(-k)\}/(1+j) \tag{8}$$

$$= H_E(k) - jH_O(k) \tag{9}$$

provided $F(k)$ and $H(k)$ are considered periodic with period N, so that $H(-k) = H(N-k)$.

The fast algorithms. If we use Equation 5 directly to calculate the DFT, we must perform N complex multiplications for each of the N values of k. Hence, the computation time is proportional to N^2. It is possible to reduce this time considerably; we call the collection of fast algorithms to compute the DFT the fast Fourier transform.[6]

Suppose that N, the number of samples, is a power of

2, say 2^p. We can rewrite Equation 5 as

$$F(k) = F_E(k) + e^{-j2\pi k/N}F_O(k) \qquad (10)$$

$$F\{k + (N/2)\} = F_E(k) - e^{-j2\pi k/N}F_O(k) \qquad (11)$$

for $k = 0, \ldots, (N/2) - 1$, where

$$F_E(k) = \sum_{n=0}^{(N/2)-1} x(2n)e^{-j2\pi \kappa n(N/2)}$$

$$F_O(k) = \sum_{n=0}^{(N/2)-1} x(2n+1)e^{-j2\pi kn/(N/2)}$$

We can see that $F_E(k)$ and $F_O(k)$ are the $N/2$-point DFTs of the even-numbered and the odd-numbered samples, respectively. The above relationship provides a way to break the original DFT into smaller transforms. We can repeat this recursive procedure until only 2-point transforms are left. We call the resulting algorithm a radix-2, decimation-in-time (DIT) FFT.

The above algorithm breaks up the transform into a binary tree pattern and thus consists of $\log_2 N$ stages. In each stage, the number of operations needed to combine pairs of transforms is proportional to N. Hence, we can complete an N-point transform in $O(N\log_2 N)$ time, which is considerably lower than $O(N^2)$. O denotes the order of $\{N\log_2 N\}$. If the Hartley transform is to compete with the Fourier transform, an $O(N\log_2 N)$ algorithm must be developed to compute the DHT.

If we substitute Equation 8 into Equations 10 and 11, we obtain (after canceling the common denominator $1 + j$):

$$\begin{aligned} H(k) + jH(N-k) = \\ [H_E(k) + jH_E\{(N/2)-k\}] \\ + e^{-j2\pi k/N}[H_O(k) + jH_O\{(N/2)-k\}] \end{aligned} \qquad (12)$$

$$\begin{aligned} H\{(N/2)+k\} + jH\{(N/2)-k\} = \\ [H_E(k) + jH_E\{(N/2)-k\}] \\ - e^{-j2\pi k/N}[H_O(k) + jH_O\{(N/2)-k\}] \end{aligned} \qquad (13)$$

for $k = 0, \ldots, (N/2) - 1$, where $H_E(k)$ and $H_O(k)$ are the $N/2$-point DHTs of the even-numbered and the odd-numbered samples, respectively. Note that we replace $-k$ by $(N-k)$ in an N-point transform and by $\{(N/2)-k\}$ in an $N/2$-point transform, because $H(k)$ is considered periodic.

We can eliminate the complex expressions in Equations 12 and 13 by equating the respective real and imaginary parts on both sides of each equation. The result is four new equations that we can combine to yield

$$\begin{aligned} H(k) = H_E(k) + [H_O(k)\cos(2\pi k/N) \\ + H_O\{(N/2)-k\}\sin(2\pi k/N)] \end{aligned} \qquad (14)$$

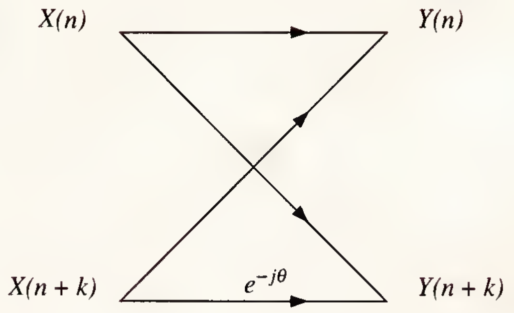


Figure 1. The butterfly structure.

$$\begin{aligned} H\{(N/2)+k\} = H_E(k) - [H_O(k)\cos(2\pi k/N) \\ + H_O\{(N/2)-k\}\sin(2\pi k/N)] \end{aligned} \qquad (15)$$

where $k = 0, \ldots, (N/2) - 1$. Equations 14 and 15 show that, like the DFT, the DHT can be recursively broken up into smaller transforms, and therefore a fast algorithm can be devised to compute it. By analogy, we call the fast algorithm a fast Hartley transform.[1]

FHT implementation

We can use the recursive formulae just discussed to implement an FHT program. However, the similarities between the FFT and the FHT, as well as the existence of numerous good FFT programs, sometimes make it simpler and more efficient to modify an existing FFT program to perform FHT. The following presents a method described by Buneman[4] to convert from FFT to FHT. This method works for any radix-2, DIT implementation.

We usually implement the radix-2, DIT FFT algorithm in the following way. We store the N input data values in an array. The program goes through p passes ($N = 2^p$), operating on the array elements in each pass; at the end of the last pass, the array contains the DFT results. To have the output values come out in the proper order, we must perform a process called bit-reversal permutation, either at the beginning or at the end of the algorithm. However, since this data permutation applies to the FFT and FHT in exactly the same way, we did not consider this in our conversion discussion.

In each of the p passes, the FFT program operates on pairs of data points in the manner dictated by Equations 10 and 11. We can summarize this operation in the structure called a *butterfly* (Figure 1). The butterfly produces two output array elements $Y(n)$ and $Y(n + k)$ from two input array elements $X(n)$ and $X(n + k)$. In the butterfly, the X's and Y's are complex (in an in-place

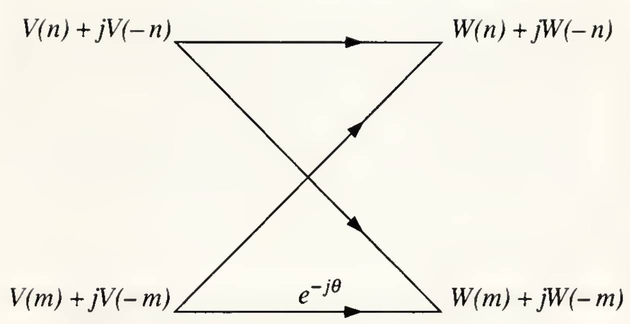


Figure 2. The FHT butterfly.

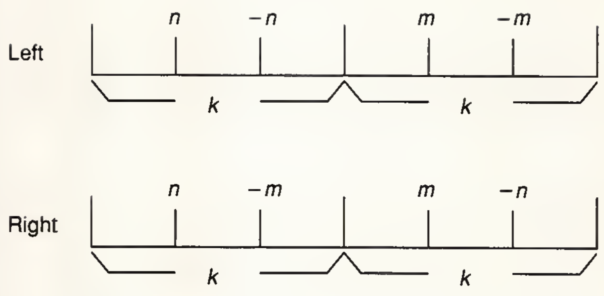


Figure 3. Interpreting the negative indices.

implementation, X and Y designate the same array). Such butterflies are collected together in groups of k consecutive butterflies. (If the passes are numbered 0 to $p-1$, then $k = 2^i$ in pass i). The first butterfly group in each pass contains butterflies corresponding to $n = 0, \ldots, k-1$.

The butterfly in Figure 1 means the following:

$$ReY(n) \leftarrow ReX(n) + \{ReX(n+k)\cos\theta + ImX(n+k)\sin\theta\} \tag{16}$$

$$ImY(n) \leftarrow ImX(n) + \{ImX(n+k)\cos\theta - ReX(n+k)\sin\theta\} \tag{17}$$

$$ReY(n+k) \leftarrow ReX(n) - \{ReX(n+k)\cos\theta + ImX(n+k)\sin\theta\} \tag{18}$$

$$ImY(n+k) \leftarrow ImX(n) - \{ImX(n+k)\cos\theta - ReX(n+k)\sin\theta\} \tag{19}$$

where θ is the phase angle of the butterfly $\{\theta = (2\pi k/N)$ for some $k\}$.

Substituting Equation 8 into the above yields the FHT butterfly (with $m = n + k$) in Figure 2. The FHT butterfly produces real outputs $W(n)$, $W(-n)$, $W(m)$, $W(-m)$ from the V inputs. This butterfly represents the following equations:

$$W(n) \leftarrow V(n) + \{V(m)\cos\theta + V(-m)\sin\theta\} \tag{20}$$

$$W(-n) \leftarrow V(-n) + \{V(-m)\cos\theta - V(m)\sin\theta\} \tag{21}$$

$$W(m) \leftarrow V(n) - \{V(m)\cos\theta + V(-m)\sin\theta\} \tag{22}$$

$$W(-m) \leftarrow V(-n) - \{V(-m)\cos\theta - V(m)\sin\theta\} \tag{23}$$

The similarities between the two sets of Equations 16 through 19 and 20 through 23 suggest the following relabeling of the array elements:

$$\begin{aligned} Re\ X(n) &\equiv V(n) \\ Im\ X(n) &\equiv V(-n) \\ Re\ X(m) &\equiv V(m) \\ Im\ X(m) &\equiv V(-m) \\ Re\ Y(n) &\equiv W(n) \\ Im\ Y(n) &\equiv W(-n) \\ Re\ Y(m) &\equiv W(m) \\ Im\ Y(m) &\equiv W(-m) \end{aligned}$$

With this relabeling scheme, Equations 16-19 and 20-23 become identical. Therefore, we can easily modify the code to compute an FFT butterfly. We can compute an FHT butterfly by reindexing the array elements used in the butterfly to implement the relabeling just described.

To resolve the negative indices, remember that the butterfly has width $k = m - n$. Hence, the signal values contained in V and W are considered periodic, with period k on the left side (V) and $2k$ on the right side (W). We interpret the negative indices according to the diagram in Figure 3. For instance, if $n < k$ (that is, the butterfly is in the first group of a pass), $-n \equiv k - n$ and $-m \equiv 2k - n$ on the left side, while $-n \equiv 2k - n$ and $-m \equiv k - n$ on the right side.

An exception must be noted in the first butterfly of each group, where the phase angle θ is zero. Since $-0 \equiv 0$, we interpret the negative indices as shown in Figure 4 on the next page.

When the phase angle is 0 or $\pi/2$, the indices $-n$ and n point to the same array element. Therefore, we must carry out only half the computations, either for the "real part" or for the "imaginary part."

This indexing scheme preserves the equations of the butterfly, but resolves the negative indices differently on the input side (left) and the output side (right). In an in-place implementation, V and W represent the same array. Thus it is more advantageous to relabel the V's and the W's in a uniform manner. For instance, in the

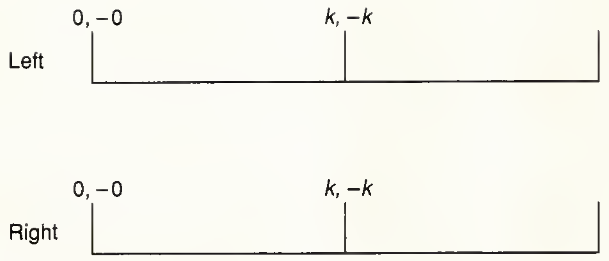


Figure 4. Interpreting the negative indices for the first butterfly of each group.

first group of a pass, we should have $-n \equiv k-n$ and $-m \equiv 2k-n$ on both sides of the butterfly. In this new scheme, we exchange the W values in Equations 21 and 23. Hence, one addition and one subtraction in those equations trade places, so that these equations no longer correspond to Equations 17 and 19 but to the following:

$$ImY(n) \leftarrow ImX(n) - \{ImX(n+k)\cos\theta - ReX(n+k)\sin\theta\} \qquad (17')$$

$$ImY(n+k) \leftarrow ImX(n) + \{ImX(n+k)\cos\theta - ReX(n+k)\sin\theta\} \qquad (19')$$

Either reindexing scheme can be used in an FFT-to-FHT conversion.

Although the FFT and FHT butterflies look very much alike, the number of butterflies to be calculated is not the same for both algorithms. In an FFT pass, each k butterfly in each group must be evaluated. On the other hand, the first $\{(k/2)+1\}$ FHT butterflies in each group cover the remaining array elements in the group as well, through the terms with negative indices. For instance, in the first group of a pass, when n runs from 0 to $k/2$, the values $\{(k/2)+1\}, \ldots, (k-1)$ will be taken care of by $k-n$ (which represents $-n$). Thus, we must compute only butterflies with phase angles in the interval $\{0, (\pi/2)\}$ in an FHT program.

Execution speed. In a group of k butterflies, we compute only $(k/2)+1$ FHT butterflies. Compare this to the previously discussed k butterflies in the FFT case.[3] Moreover, two of these butterflies correspond to phase angles of 0 and $\pi/2$, and contain only half as many operations as do their FFT counterparts. Therefore, we can compute a group of FHT butterflies twice as fast as a group of FFT butterflies. It follows that the FHT execution time is only half the FFT execution time.

An FFT program written in assembly language for the IBM PC and converted to an FHT program in the manner described earlier yielded timing results that confirmed the theoretical prediction. A 1,024-point FFT executed in 71.23 milliseconds on a 10-MHz IBM PC AT, whereas a 1,024-point FHT took 35.78 ms on the same machine for an FFT-to-FHT ratio of 1.99.

It should be remembered that an FHT is twice as fast as a *complex-input* FFT, but no faster than a *real-input* FFT based on the Hermitian property. The speed advantage of the FHT is only apparent in the inverse transformation, in which we must use a complex FFT in any case.

Memory storage. The Hartley transform yields real outputs, while the Fourier transform yields complex outputs. Hence, an FHT program requires only half the array space of an FFT program, for the same size of transform. Also, the FHT algorithm only needs cosine and sine values for angles from 0 to $(\pi/2)$, while the FFT needs angles up to π. This represents an extra saving in memory space for programs that keep cosine and sine values in tables.

Usefulness in applications. The two most important and widely used applications of the FFT in digital signal processing are power spectrum calculation and convolution. We can perform these computations with the FHT as easily as with the FFT.

The power spectrum of a signal having Fourier transform F is given by

$$P = (ReF)^2 + (ImF)^2$$

Substituting Equation 9 into this yields

$$P(k) = \{H(k)^2 + H(-k)^2\}/2$$

where $H(k)$ is the Hartley transform of the signal. Thus, we obtain the power spectrum directly from the FHT outputs and with the same speed as when the FFT is used.

Two power spectrum programs, one using the FFT and the other using the FHT for 1,024 data points, yielded the following timing results on a 10-MHz IBM PC AT. The time spent on calculating the power spectrum from the transformed data was 31,768 cycles for the FFT and 34,830 cycles for the FHT. Since a 1,024-point FHT executes on the machine in 35.8 ms (as reported earlier), and a real-input FFT has the same speed, it follows that the total computation time was approximately 39 ms for the FFT and 39.3 ms for the FHT, about the same for both transforms.

Consider the simple linear system displayed in Figure 5. The output signal $g(t)$ is given by

$$g(t) = f(t) * h(t)$$
$$= F^{-1}\{G(\omega)\}$$

where

$$G(\omega) = H(\omega)F(\omega);$$

$$F(\omega) = F\{f(t)\};$$

$$H(\omega) = F\{h(t)\};$$

F and F^{-1} denote the Fourier transform and its inverse, respectively, and $*$ denotes convolution. The derivation of $g(t)$ requires two Fourier transforms $f(t)$ and $h(t)$ and one inverse Fourier transform $G(\omega)$.

If two signals $x_1(n)$ and $x_2(n)$ are convolved to yield $x_3(n) = x_1(n) * x_2(n)$, the Fourier transform of $x_3(n)$ becomes $F_3(k) = F_1(k)F_2(k)$. Here, $F_1(k)$ and $F_2(k)$ are the Fourier transforms of $x_1(n)$ and $x_2(n)$, respectively. Replacing the real and imaginary parts of the Fourier transforms with the even and odd parts of the Hartley transforms as in Equation 9, we can show that[1]

$$H_3(k) = 1/2\ \{H_1(k)H_2(k) - H_1(-k)H_2(-k) + H_1(k)H_2(-k) + H_1(-k)H_2(k)\} \qquad (24)$$
$$= H_1(k)H_{2E}(k) + H_1(-k)H_{2O}(k)$$
$$= H_2(k)H_{1E}(k) + H_2(-k)H_{1O}(k) \qquad (25)$$

where the H's are the Hartley transforms of the corresponding signals, and the subscripts E and O denote the even and odd parts, respectively. Although the convolution formula for the Hartley transform looks more complicated than the one for the Fourier transform, it consists of the same number of real multiplications as does the complex multiplication in the other formula, and only one extra real addition. Moreover, in many applications, the signals are usually even or odd. In these cases, one half of Equation 24 or Equation 25 vanishes, making it possible to perform convolution with one real multiplication of the Hartley transforms.[1,3]

Convolution provides a good means to compare the FFT and the FHT in applications, since it involves both the forward and the inverse transformations. As an example, consider a discrete version of the system shown in Figure 5, with two forward and one inverse transformations. Assume that the computation time for a general DFT is T_{DFT}, and that a multiplication in frequency domain is T_{MUL}. The computation time for the output $g(n)$ is thus $3T_{DFT} + T_{MUL}$, in general. If the input signal is real and the Hermitian property is used to reduce the forward DFT time to $0.5T_{DFT}$, we can compute $g(n)$ in $2T_{DFT} + T_{MUL}$. Usually $T_{MUL} \ll T_{DFT}$. Hence, the Hermitian property allows us to reduce the convolution computation time by a 2/3 factor.

If we use the DHT in the previous example, we can reduce the computation to $0.5T_{DFT}$ in both the forward and inverse transformations, resulting in a total computation time of $1.5T_{DFT} + T_{MUL}$. This time is approximately one half of that for the general (complex-input) DFT and three quarters of that for the real-input DFT.

Two convolution programs, one using the FFT and the other using the FHT for 1,024 data points, yielded the following timing results on a 10-MHz IBM PC AT. Multiplying the transforms of the input signals took 82,577 cycles for the FFT version and 70,083 cycles for the other. The FHT computations (forward and inverse), as well as the *forward* FFT computations, took about 35.8 ms; the *inverse* FFT time was about 71.2 ms. Since there are two forward and one inverse transformations, the total execution time was about 114.4 ms for FHT and 151.1 ms for FFT. This gave us an FHT-to-FFT ratio of 0.76, which is very close to the estimate of 0.75 given earlier.

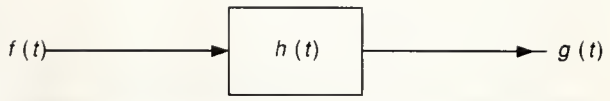


Figure 5. A simple linear system.

It seems disappointing that the FHT does *not* reduce the execution time by half. However, remember that both the forward and the inverse FHT call for the same routine. In addition, the FFT must use two separate routines for the forward and inverse transforms if real input signals are to be treated differently, rendering the program more complicated. On the other hand, an FFT program with the capability to switch automatically between real-input and complex-input versions would be more general than an FHT program, since it would be able to handle any kind of signals.

Hardware implementation. We based a paper design on an existing FFT processor to see how easy it would be to switch from the FFT to the FHT in hardware. We arbitrarily chose the 6-cycle butterfly FFT processor presented by R.J. Karwoski[7] for this job. It turned out that hardware components and structures designed for FFT can be used to implement FHT with only minor modifications.

A cycle count reveals that the FHT processor would take exactly half as many clock cycles as its FFT ancestor to complete a transformation. The FFT processor computes each of the $(N/2)\log_2 N$ butterflies in six cycles. The FFT took $3N\log_2 N$ cycles, and the FHT thus took $1.5N\log_2 N$ cycles. Since we used a clock period of 100 nanoseconds, a 1,024-point FFT took 3.07 ms and a 1,024-point FHT took 1.54 ms.

Comparative performance on various systems. Table 1 on the next page lists a survey of the execution times of many FFT and FHT software and hardware implementations, to give an idea of the performance attainable with the present technology. For the FFT we include some timing values that are actual measurements performed at Concordia University and some from benchmarks obtained from other sources. Only FHT values for the Intel 86 microprocessor family reflect actual measurements. We took the others to be one half of the correspondent FFT values, since Bracewell[1,3] has proven this in theory, and we have shown it in practice here to be the case.

Table 1.
The 1,024-point FFT and FHT execution times.

Manufacturer	Processor	Clock (MHz)	Complex FFT time (ms)	FHT time (ms)
Software				
Intel Corp.	8086*	8	292.8	144.6
	80286*	10	71.2	35.8
	80386*	16	43.1	21.4
Texas Instruments	TMS32020**	5	31.8	15.9
	TMS320C25†	10	13.2	6.6
	TMS320C30‡	16.7	3.75	1.88
Analog Devices	ADSP-2100†	8	7.6	3.8
National Semiconductor	LM32900†	10	13.4	6.7
NEC	PD77230†	6.7	10.8	5.4
Phillips/Signetics	DSP1†	8	6.85	3.43
Motorola	DSP56000§	10	4.99	2.5
Hardware				
	Design based on Karwoski[7]	10	3.07	1.54
	Sequential 100-ns butterflies¶		0.512	0.256
	Cascaded 100-ns butterflies¶		0.0512	0.0256
	Parallel 100-ns butterflies¶		0.001	0.0005
	Array 100-ns butterflies¶		0.0001	0.00005

*Obtained from programs developed at Concordia University
**Based on results in Papamichalis and So.[8]
†Based on results in Marrin.[9]
‡Based on results in Texas Instruments.[10]
§Based on results in Morris.[11]
¶Based on results in Mavor and Grant.[12]

As just shown in the case of power spectrum calculation and convolution, the Hartley transform becomes an adequate substitute for the popular Fourier transform in general applications involving real signals. If the Fourier transform is explicitly required, we can always obtain it from the Hartley transform with Equation 8 or Equation 9. But usually we can use the Hartley transform directly.

The FHT is very easy to implement based on existing FFT software or hardware components. It can also benefit from improved techniques intended for the FFT, such as new algorithms to calculate bit-reverse indices[13] or faster hardware multipliers.

The FHT, generally twice as fast as the complex-input FFT, only has the same speed as the real-input FFT. However, even if the original input time signal is real, the inverse FFT still must deal with complex-input values. Thus, the FHT indeed proves faster than the FFT when both the forward and the inverse transformations are required—such as in convolution—although not quite twice as fast. An FFT program that can take advantage of real input signals and is also capable of handling complex inputs, while perhaps more general than an FHT program, will certainly be larger and much more complicated.

References

1. R.N. Bracewell, *The Hartley Transform*, Oxford University Press, New York, 1986.
2. R.V.L. Hartley, "A More Symmetrical Fourier Analysis Applied to Transmission Problems," *Proc. IRE*, Vol. 30, Mar. 1942, pp. 144-150.
3. R.N. Bracewell, "The Fast Hartley Transform," *Proc. IEEE*, Vol. 72, No. 8, Aug. 1984, pp. 1010-1018.
4. O. Buneman, "Conversion of FFTs to Fast Hartley

Transform," *SIAM J. Sci. Stat. Computing,* Vol. 7, No. 2, Apr. 1986, pp. 624-638.

5. R.N. Bracewell, *The Fourier Transform and Its Applications,* 2nd ed., McGraw-Hill, New York, 1965.
6. E.O. Brigham, *The Fast Fourier Transform,* Prentice Hall, Englewood Cliffs, N.J., 1974.
7. R.J. Karwoski, "An Introduction to Digital Spectrum Analysis Including a High Speed FFT Processor Design," Application Note TP10A-7/81, TRW LSI Products, Calif., June 1980.
8. P. Papamichalis and J. So, "Implementation of Fast Fourier Transform Algorithms with the TMS32020," *Digital Signal Processing Applications with the TMS320 Family,* Texas Instruments, Dallas, 1986, pp. 69-168.
9. K. Marrin, "Six DSP Processors Tackle High-End Signal-Processing Applications," *Computer Design,* Mar. 1986, pp. 21-25.
10. Texas Instruments, "TMS320C30 FFT Benchmarks," *Details on Signal Processing,* Issue 12, Sept. 1987.
11. L.R. Morris, "Digital Signal Processing Microprocessors: Forward to the Past?," *IEEE Micro,* Vol. 6, No. 6, Dec. 1986, pp. 6-8.
12. J. Mavor and P.M. Grant, "Operating Principles and Recent Developments in Analogue and Digital Signal Processing Hardware," *Proc. IEEE,* Vol. 134, No. 4, July 1987, pp. 305-328.
13. D.M.W. Evans, "An Improved Digit-Reversal Permutation Algorithm for the Fast Fourier and Hartley Transforms," *IEEE Trans. Acoustics, Speech, and Signal Processing,* Vol. ASSP-35, No. 8, IEEE CS Press, Los Alamitos, Calif., Aug. 1987, pp. 1120-1125.

Tho Le-Ngoc is an associate professor of electrical and computer engineering at Concordia University, Montreal. He has worked in the design and development of satellite and terrestrial microwave communications equipment with Spar Aerospace Limited and SR Telecom Inc. He has also served as a consultant to several industries. His research areas of interest include digital communications and signal processing.

Le-Ngoc obtained his BEng and MEng degrees from Montreal's McGill University and his PhD from the University of Ottawa.

Minh Tue Vo is a final-year undergraduate student in the Electrical Engineering Department of the University of Waterloo in Ontario. His current field of interest is digital signal processing.

Questions concerning this article can be addressed to Le-Ngoc at the Department of Electrical and Computer Engineering, Concordia University, 1455 de Maisonneuve West, Montreal, Quebec, Canada H3G 1M8.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 156 **Medium** 157 **High** 158

A Benchmark Tutorial

What do benchmark test results really mean? Here's how to evaluate the data in terms of your own application.

Walter J. Price
Motorola

Simply put, a benchmark is a standard for judging relative performance among various computers. Unfortunately, any simplicity ends with this definition. The first problem arises from the fact that users have no official standard to follow when they want to evaluate a benchmark. The second problem revolves around the truism that the best benchmark in the world that measures someone else's application does just that. From the user's perspective, the best benchmark in the world accurately measures system performance in a target application. The task of creating a good benchmark includes the determination of which tests are applicable to the user's environment and how to determine the results.

Here I generally describe benchmarking and discuss some of the specific benchmark tests for computer systems that are in use today. I also examine some of the pitfalls involved with benchmark comparison and analysis and point out how to avoid them—or at least to minimize the impact of such problems. The goal is to learn how to gather and interpret meaningful comparison data.

Benchmarking caveats

Producing and interpreting benchmark data crosses into the realm between art and science. No single-task benchmark test (one that only measures one aspect of a computer's performance) can fully characterize the true performance capability of a system under actual user loads. Safety does reside in numbers. A collection of different benchmark tests merely provides some averages. But users must take care when comparing the results.

Features like optimizing options on today's compilers can drastically affect the results of benchmark tests. Some vendors publish a range of test results using different optimization levels. This documentation probably provides the most reliable form of public-domain benchmark data. Combining this data with the appropriate system configurations that were actually used in the tests provides the background information needed to reproduce the data, if necessary.

0272-1732/89/1000-0028$1.00 © 1989 IEEE

Users should also consider the source or "pedigree" of a benchmark. Benchmark code tends to migrate quite freely, given public-domain networks like Usenet. Modifications made to this code, for whatever reason, cause the evolution of many different versions. Hand-coded libraries or conversions from one language to another can cause successive generations of the test. Results vary according to which version of a benchmark test is used. Because of this process, some benchmarks have degenerated to the point of being virtually useless.

Users should avoid making a determination of overall system performance based on one benchmark test. Many benchmarks are one-dimensional in nature (that is, they test only one aspect of a system). Some popular tests stress only raw processor-instruction bandwidth or floating-point performance. Other items affecting system performance include

- file system and compiler efficiencies,
- cache-memory size and organization,
- main-memory size and organization,
- I/O and file transfer speeds,
- user loads, and
- application mixes.

Most benchmark tests do not verify the validity of the performance results they produce. Without verified test results, users can find it difficult to determine whether the benchmark was properly performed. Some optimizing compilers recognize small benchmark suites and simply return the expected result without performing any computation. Totally meaningless performance data results.

The only way to truly compare benchmark test results is to port and compile the same copy of the operating-system source to all of the systems under consideration. This procedure minimizes the effects from differences in binary copies of the code and ensures a common base of software for these systems. Since this task is beyond the reach of most organizations, users can employ another option. Setting up the test carefully and obtaining a clear understanding of how to interpret the results can yield a successful comparison. The goal is to establish a level playing field for all of the systems being tested.

Some additional questions to ask when analyzing or generating benchmark data follow.

1) Which real-world application does the benchmark measure, and how does the user accurately characterize the system's performance under typical application work loads?

Try to choose a set of benchmark tests that analyzes the components critical to producing optimal user performance. For example, a CPU-intensive benchmark by itself does not provide sufficient data to interpret how a system will perform in an I/O-intensive environment like transaction processing. In this example, the benchmark suite should also measure disk-I/O and file-system efficiencies and work-load capacity (keystroke-handling capability).

2) What factors influenced the generation of the benchmark data?

Correlating data from different or biased sources probably will not produce an accurate comparison. A vendor may provide data from a "hot box" (or nonproduction hardware) or use a compiler that "recognizes" a benchmark suite and loads a hand-optimized algorithm for the test. Valid data should contain documentation of test configurations and optimization levels. The tests should be repeatable using a production-grade system.

The benchmark test-result data that appears in this article (shown in tables) constitutes a general guideline only. Users should implement application-specific tests (that is, actual application software running in a typical user environment) to supplement this information.

3) Were devices like cache accelerators or optimizing compilers used in all of the configurations tested?

Cache accelerators can make a small benchmark run much faster than if the same code were to run in another memory location. Optimizing compilers can virtually reduce a repetitive benchmark to a simple NOP (no operation) and yield totally meaningless results.

4) Was one system tuned for a particular benchmark?

Some vendors have a reputation for publishing benchmark data obtained from a hot box or a modified system that a customer could not buy or duplicate with off-the-shelf components. Simple operating-system kernel tuning can improve benchmark results by as much as 50 percent. Hand-optimized utilities can improve benchmark performance by as much as 30 percent. All operating-system parameters and compiler options should match closely from one system to another.

5) What sources should users employ for benchmark data?

To assure the best comparison of results, users should investigate several different sources. When possible, they should use the results from the same test suite on all systems being evaluated. Users can also employ independent third-party benchmark data as a verified, unbiased source of information.

In summary, users should

- determine which aspects of system or component performance are to be measured,
- determine the best source of benchmark suites or

performance data (either public-domain or licensed third-party packages),

• ensure that all system-hardware and operating-system parameters during benchmark comparisons equate as closely as possible, and

• understand what specific benchmark tests measure and what causes the results to vary.

Benchmark tests

Here I present a set of descriptions for many of the more popular benchmarks. I obtained the actual benchmark data contained in the following tables from a variety of public-domain sources. I do not intend these benchmark figures to provide definitive results, but show them to give a general indication of relative performance among various computers. In many instances, a published range of performance values exists for various computers. Since space cannot reasonably present the total data, I list only the highest and lowest benchmark results for a system in terms of its particular measurement.

I show the system model number and the general test configuration (processor type, the typical amount of cache and random-access memory available, the processor speed in megahertz, and the processor rating in millions of instructions per second) for each entry. (See the accompanying box for a discussion of MIPS.) This data provides a basic understanding of the test setup. As stated, obtaining a description of the entire hardware and software test setup—along with the actual results—adds meaning to model-by-model comparisons.

Finally, some table entries contain additional information in the comments column that provides background data on how test results were obtained. Examples include the operating system (such as VMS for the VAX and AIX for the IBM computer), the compiler options (optimized, unoptimized), and any special hardware (68882 floating-point units, or FPUs, and floating-point accelerators, or FPAs) used in the test. Dashes indicate the information was not published with the test results, or is generally unavailable.

The industry uses a number of public-domain or licensed third-party benchmarks, some of which I discuss in the following sections. (For an outline of some popular proprietary benchmarks, see the accompanying box.)

Dhrystone. This synthetic (nonreal-world) benchmark measures processor and compiler efficiency by executing a "typical" set of integer calculations. These calculations include integer arithmetic, character/string/array manipulation, and pointers. Reinhold P. Weicker constructed the benchmark by using measured statistical data from actual user programs. The procedure does not use operating-system calls, I/O functions, or floating-point operations. The results provide a general measure of user-level integer performance. This benchmark contains little code that can be optimized by vector processor systems.[2]

Weicker wrote the original Dhrystone benchmark in the Ada programming language. Rick Richardson later rewrote it in the C language and posted it on the Usenet network. Results appear in Dhrystones per second. Users should exercise caution with the Dhrystone test (as with any benchmark test) because it does not always reflect how large user applications perform. Also, optimizing compilers can remove useless code from Version 1.1 of the benchmark, which improves the per-

MIPS: A meaningful measurement?

Millions of instructions per second, or MIPS, is a popular—though superficial—way to describe computer system performance. MIPS typically come from either measured data or calculated maximum performance. We usually compare system performance ratings in MIPS with that of a VAX 11/780, which is considered to be a 1-MIPS machine. On a family of systems with a common processor (like the Motorola 88000), MIPS ratings can help judge relative system integer performance. The ultimate apples-to-oranges phenomenon occurs when we compare MIPS ratings between two different architectures like RISCs and CISCs (reduced and complex instruction-set computers). The key points to remember about MIPS are

• instructions do not remain constant from processor to processor, and

• MIPS are only meaningful in the context of a single processor family.

To make sense out of MIPS ratings, users must first define the term *instruction*. A direct correlation does not always exist between the number of instructions being executed and the amount of actual work being accomplished. On a jovial note, the industry has come up with many colorful ways to use the MIPS acronym:

• meaningless indicators of performance for systems,

• meaningless information of performance for salesmen, or

• meaningless information from pushy salesmen.

Proprietary benchmarks

The following performance benchmarks—in contrast to the others in this article—have not entered the public domain. Users who wish to obtain more information should contact the companies directly.

Aim benchmarks. Aim Technology in Palo Alto, California, sells and maintains two suites of multiuser benchmark tests.

Suite III. Written in the C programming language, this suite simulates applications that fall into either task- or device-specific categories.

The task-specific routines simulate such functions as word processing, database management, and accounting. The device-specific code measures the performance of hardware features like memory, disk, floating-point, and I/O operations. All measurements represent a percentage of VAX 11/780 performance.

The performance and user ratings constitute the two most frequently quoted results. The performance rating represents a percentage of VAX 11/780 performance in which the VAX equals 100 percent (for example, the Motorola SYS3640 supermicrocomputer has a performance rating that is 400 percent of the VAX, or four times the performance).

The second parameter, or user rating, is the maximum number of concurrent users that a system can support. For example, a VAX 11/780 equals 12 users. In general, the Aim III suite gives a better overall indication of a system's performance than small, single-task benchmarks. The company verifies and maintains all official results. Published, copyrighted reports are available on individual computer systems.

Suite V. This benchmark suite measures throughput in a multitasking workstation environment. The design goals of this new suite include

- the ability to stress single-user, multitasking system performance,
- simulations of real-world systems that employ routines based on actual user applications,
- incremental system loading to gradually increase the stress on system resources, and
- testing multiple aspects of system performance.[1]

The graphically displayed results plot the workload level versus the amount of time (in seconds) to process the specific load level. Several different models characterize various user environments such as financial, publishing, and software development.

Business Benchmark. Developed and maintained by Neal Nelson and Associates in Chicago, this collection of 18 separate routines examines various aspects of system performance. The bulk of these routines consists of a series of loops that exercise functions such as disk I/O speed, floating-point performance, and processor and cache efficiency. Various combinations of the 18 routines characterize different real-world business applications like word processing, accounting, and application development. Unlike many other benchmark tests in use today, Business Benchmark results indicate that CISCs tend to outperform RISCs on some multiuser tasks.

formance results by as much as a factor of two. Dhrystone Versions 2.X eliminated this dead code to thwart the efforts of current optimizing compilers. Table 1 on the next page reflects Version 1.1.

Originally, Dhrystone solely used features like peephole optimizers, but today anything appears to be acceptable—short of in-line coding. When benchmark numbers are quoted, users should ask which compiler options were selected at execution time.

The Dhrystone routine itself contains some peculiar attributes that have been coded into the program. Part of the routine involves copying long, 30-character strings that happen to be on unaligned word boundaries. Dhrystone uses zero-offset address for 50 percent of its memory data references, where a more real-world number is somewhere between 10 and 15 percent.[2] These attributes tend to make some machines appear faster than others. The Dhrystone test seems to be lenient on processors like the Am29000 because the routine does not exercise CPU areas that would slow the 29000 down under typical user-application loads.[3] The Dhrystone benchmark is also small enough to fit into the instruction cache of some systems, which further skews the results.

Table 1.
Summary of Dhrystone 1.1 benchmark test results.

		Cache	RAM	Frequency	MIPS	Dhrystones/s		
System/model	Processor	(Kbytes)	(Mbytes)	(MHz)	rating	Low	High	Comments
Alliant FX/8; MP=8	Proprietary	—	—	—	35.0	7,655	7,655	
ALR FlexCache 25386	80386	64	5	25.00	—	8,671	8,671	SCO Unix
Altos Series 2000	80386	—	—	16.00	3.0	4,237	4,348	
Amdahl 5860	—	—	—	—	—	28,846	28,846	C compiler, V. 1.22
Amdahl 5890/300E	—	—	—	—	—	43,668	43,668	C compiler
Apollo 5X0T	68020	—	—	20.00	3.4	6,250	6,250	
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	25,461	27,000	Up to four CPUs
Apollo Series DN3000	68020	—	—	12.50	1.2	2,186	2,186	
Apollo Series DN4000	68020	0	4	25.00	4.0	6,038	7,109	
Apple Macintosh	68000	—	—	7.70	—	625	625	
Apple Macintosh II	68020	—	—	15.70	—	2,106	2,719	H = Greenhills C compiler V. 1.8
Apple Macintosh Plus	68000	—	—	7.83	—	660	769	
AT&T 3B1	68010	—	—	10.00	—	973	1,033	Unix V. 1
AT&T 3B2/300	32000	—	—	7.20	—	409	699	L = Unix V. 2; H = Unix V. 3
AT&T 3B2/400	32100	6	4	10.00	1.0	672	1,120	L = Unix V. 3; H = Unix V. 2
CCI Power 5/32	68010	—	—	12.50	—	1,129	1,192	Unix 4.2 BSD
CCI Power 6/32	Proprietary	—	—	—	—	8,498	8,498	
CCI Power 7/64	Proprietary	—	—	—	—	53,108	53,108	
Compaq 386	80386	0	4	16.00	—	1,724	2,941	
Compaq 386/20	80386	0	4	20.00	—	7,575	9,335	
Compaq 386/25	80386	32	5	25.00	—	8,277	10,617	
Convergent Server PC	80386	64	4	20.00	5.7	6,534	9,436	L = unopt.; H = opt.
Convex C-1 XP 6.0	Proprietary	—	—	—	4.5	7,249	7,249	
Cray 1S	Proprietary	—	—	—	—	14,820	14,820	
Cray X-MP	Proprietary	—	—	—	—	18,530	18,530	
Data Gen. MV15000-20	Proprietary	16	64	11.80	8.0	8,300	8,300	
Data Gen. MV20000	Proprietary	16	64	—	10.0	8,300	8,300	
DEC μVAX 3500	Proprietary	—	—	11.10	3.5	4,746	4,746	
DEC μVAX II	Proprietary	—	—	5.00	0.9	1,326	1,612	L = VMS
DEC VAX 11/750	Proprietary	—	—	—	1.0	835	961	
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	1,243	1,870	
DEC VAX 11/784	Proprietary	—	—	—	—	5,263	5,555	
DEC VAX 11/785	Proprietary	—	—	7.50	1.5	1,783	2,069	
DEC VAX 8550	Proprietary	—	—	—	6.4	8,000	10,416	
DEC VAX 8600	Proprietary	—	—	—	4.5	6,423	10,416	L = Unix 4.3 BSD
DEC VAX 8600	Proprietary	—	—	—	4.5	4,896	5,235	Ultrix V. 1.2
DEC VAX 8650	Proprietary	—	—	—	6.2	7,123	10,787	H = VMS
DEC VAX 8700	Proprietary	—	—	—	6.0	10,416	10,416	
DEC VAX 8810	Proprietary	—	—	22.22	12.0	10,416	10,416	
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	1,502	1,502	
DEC Vaxstation 3200	Proprietary	—	—	5.00	2.0	5,271	5,271	
Decstation 3100	R2000	128	8	16.70	14.0	16,870	26,600	L = unopt.; H = opt.
Force CPU-21B	68020	—	—	25.00	—	5,555	5,555	
Force CPU-386A	80386	—	—	16.00	—	6,200	6,200	
HP 9000 Mod. 320	68020	—	—	16.70	2.0	2,464	2,671	HP/UX V. 5.02
HP 9000 Mod. 340	68030	—	—	16.70	—	6,536	6,536	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	6,702	6,702	
HP 9000 Mod. 500	Proprietary	—	—	—	—	1,599	1,599	HP/UX V. 5.05; 1 processor
HP 9000 Mod. 500	Proprietary	—	—	—	—	3,020	3,020	HP/UX V. 5.05; 2 processors
HP 9000 Mod. 500	Proprietary	—	—	—	—	4,140	4,140	HP/UX V. 5.05; 3 processors
HP 9000 Mod. 550	Proprietary	—	—	—	—	1,518	1,531	HP/UX V. 5.11
HP 9000 Mod. 825S	Prop. RISC	16	—	12.50	3.0	17,829	17,829	
HP 9000 Mod. 825SRX	Prop. RISC	16	—	—	8.0	13,157	16,672	
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	23,430	23,441	
HP 9000 Mod. 835SRX	Prop. RISC	128	—	15.00	4.0	23,430	23,430	
HP 9000 Mod. 840	Prop. RISC	128	24	8.00	4.5	11,165	11,215	H = opt.

Table 1 (continued).

		Cache	RAM	Frequency	MIPS	Dhrystones/s		
System/model	Processor	(Kbytes)	(Mbytes)	(MHz)	rating	Low	High	Comments
HP 9000 Mod. 840S	Prop. RISC	128	24	8.00	4.5	9,920	9,920	
HP 9000 Mod. 850S	Prop. RISC	—	—	13.70	7.0	15,576	21,358	
IBM 3081	—	—	—	—	—	15,007	15,007	C compiler V. 1.5
IBM 3090/200	—	—	—	—	10.0	31,250	31,250	
IBM 4341 Mod. 12	Proprietary	—	—	14.70	—	3,690	3,910	Opt.
IBM 4381 Mod. 2	Proprietary	—	—	—	—	4,504	5,681	
IBM PC AT	80286	—	—	—	—	1,380	1,380	
IBM PC AT	80286	—	—	6.00	—	531	531	
IBM PC AT	80286	—	—	9.05	—	692	1,484	
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	8,650	12,769	L = SCO Unix ; H = AIX
IBM RT PC	Prop. RISC	—	—	5.90	4.5	6,097	6,500	H = AIX
IBM RT PC Mod. 135	Prop. RISC	—	—	7.30	6.0	10,770	10,770	
Integr. Sol. Advantage2000	R2000	64	16	16.70	12.0	18,920	27,100	L = unopt.; H = opt.
Intel 386 ATS	80386	—	—	16.00	—	3,424	3,424	
Intergraph Interpro 32C	Clipper RISC	—	—	30.00	5.0	4,855	8,309	
Ironics IV-9001	Am29000	16	8	25.00	17.0	35,760	35,760	
MIPS M/1000	R2000	128	16	15.00	15.0	15,100	25,000	L = unopt.; H = opt.
MIPS M/120-3	R2000	128	8	12.50	10.0	23,300	23,300	
MIPS M/120-5	R2000	128	8	16.70	13.0	18,700	31,000	L = unopt.; H = opt.
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	30,700	47,400	L = unopt.; H = opt.
MIPS M/500	R2000	24	8	8.00	8.0	8,800	14,200	L = unopt.; H = opt.
MIPS M/800	R2000	128	8	12.50	12.5	12,800	21,300	L = unopt.; H = opt.
MIPS RC2030	R2000	64	16	16.70	12.0	19,100	31,200	L = unopt.; H = opt.
Motorola SYS1131	68020	16	2	16.70	1.5	3,246	3,257	
Motorola SYS1147	68030	0	4	20.00	3.8	6,334	6,334	
Motorola SYS2300	68020	0	4	16.70	1.5	4,876	4,876	
Motorola SYS2600	68020	16	4	16.70	1.5	4,566	4,566	
Motorola SYS3300	68030	0	4	20.00	3.8	6,334	6,334	
Motorola SYS3600	68030	0	4	25.00	4.7	8,826	8,826	
Motorola SYS3640	68030	64	4	25.00	5.3	7,942	8,900	
Motorola SYS3800	68030	64	—	33.00	6.9	9,239	11,000	H = Greenhills C V. 1.8.2
Motorola SYS8600	88100	32	8	20.00	17.0	35,714	35,714	
Multiflow Trace 7/200	—	—	—	—	—	14,195	14,195	
NCR Tower 32/400	68020	8	4	16.70	1.5	3,628	3,638	
NCR Tower 32/450	68020	8	4	25.00	—	4,941	4,941	
Opus Systems	88000	32	4-20	20.00	17.0	41,166	41,166	
Prime EXL 316	80386	—	—	16.00	3.2	7,112	7,112	Optimized
Pyramid 90x	Proprietary	—	—	8.00	2.5	1,779	3,333	High = w/cache
Pyramid 98x	Proprietary	—	—	10.00	5.4	3,627	3,856	
Silicon Graphics Iris	R2010	24	8	12.50	—	18,416	18,416	
Solbourne 4/600	Sparc	64	16	16.70	7.0	18,715	18,715	
Sparcstation 1	Sparc	0	4-24	20.00	12.5	22,049	22,049	
Sparcstation 330	Sparc	0	—	25.00	16.0	27,777	27,777	
Sun 3/160C w/68881	68020	0	4	16.70	2.0	2,800	3,850	L = unopt.; H = opt.
Sun 3/260 w/68881	68020	0	8-24	25.00	4.0	5,366	7,142	
Sun 3/470	68030	0	—	33.00	7.0	11,748	11,748	
Sun 3/50	68020	0	4	15.00	1.5	2,280	2,695	
Sun 3/60	68020	0	4	16.70	2.0	4,295	4,545	
Sun 3/80	68030	0	4	20.00	3.0	5,154	5,154	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	8,388	8,388	w/80387 FPU
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	10,550	19,900	L = unopt.; H = opt.
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	14,109	14,201	
Tandy 3000	80286	—	—	8.00	—	1,455	1,543	
Tandy 6000	68000	—	—	8.00	—	1,286	1,366	
Tektronix 4319	68020	0	4	20.00	2.5	7,581	7,581	
Unisys 5000/90	68020	—	—	12.50	1.0	3,307	3,311	

Table 2.
Summary of *Digital Review* benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Seconds Low	Seconds High	Comments
Alliant FX/8	Proprietary	—	—	—	35.0	1.48	1.48	
Convex C-1 XP	Proprietary	—	—	—	4.5	0.487	0.487	
DEC μVAX II/GPX	Proprietary	—	—	5.00	0.9	9.17	9.17	
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	6.75	6.75	
DEC VAX 8600	Proprietary	—	—	—	4.5	2.32	2.32	VMS V. 4.5
DEC VAX 8650	Proprietary	—	—	—	6.2	1.584	1.584	
DEC VAX 8700	Proprietary	—	—	—	6.0	1.469	1.469	
DEC Vaxstation 3200	Proprietary	—	—	5.00	2.0	2.9	2.9	
Elxsi 6420	Proprietary	—	16	20.00	—	1.193	1.193	
MIPS M/1000	R2000	128	16	15.00	15.0	0.94	0.99	
MIPS M/120-5	R2000	128	8	16.70	13.0	0.783	0.783	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	0.553	0.553	
MIPS M/500	R2000	24	8	8.00	8.0	1.86	1.86	
MIPS M/800	R2000	128	8	12.50	12.5	1.2	1.2	
Sun 4/260	Sparc	0	16-32	16.70	10.0	1.72	2.09	Weitek FPU
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	2.32	2.32	

Digital Review. *Digital Review* magazine has compiled a set of benchmark routines that mixes 34 individual integer and floating-point routines. The test itself stresses floating-point performance. This large benchmark contains over 3,000 lines of Fortran code. The *Digital Review* benchmark does not perform any verification of test results; these results usually appear as a list of the geometric mean of all tests, performed in seconds. On a secondary level, the test normalizes relative comparisons among various systems to the Digital MicroVAX II, which is equal to 1.0. These units are called MicroVAX units of processing (MVUPs). Table 2 lists only the raw benchmark-test results in seconds.

Users have criticized this benchmark for its odd structure and unusual instruction mix that does not accurately mimic real-world program flow. Initializing the routines within the timing loops, rather than running the actual benchmark code, consumes a large amount of time. This practice usually results in a low estimate of a system's actual MVUP rating. More recently, *Digital Review* magazine has taken steps to revise its benchmark (now called CPU2) to correct some of these odd programming sequences. (Table 2 reflects the previous version.)

Dodoc. This 5,300-line Fortran program—which simulates the operations within a nuclear reactor—accurately tests instruction-fetch bandwidth and scalar floating-point performance. Compilers can vectorize very little of the code. The routine uses the Monte Carlo method of simulation in which an iterative process converges on an expected result. The routine was originally designed as a check of both compiler and intrinsic (real-world) functions. Normalized results appear in terms of the ratio of CPU time needed to perform the test versus an arbitrary defined reference.[4] This *R* factor is normalized where 100 equals the performance of the IBM 370 Model 168. The algorithm calculates that $R = 48{,}671/\text{seconds}$ of processor time. Larger *R* factors equate to higher system performance.

Differences in floating-point accuracy (that is, single- or double-precision calculations), unique characteristics of mathematical libraries, and rounding errors contribute to how fast the algorithm converges on the expected answer. (See Table 3.)

Khornerstone. Developed by Workstation Laboratories, this benchmark yields a normalized rating on overall system performance using 22 separate tests.[5] This suite of tests includes a mix of both public-domain (Dhrystone, Sieve, etc.) and proprietary benchmark routines. The result is a unit of measure called Khornerstones per second. This set of routines measures characteristics of processor, floating-point, and disk performance. The Khornerstone test measures single-user

Table 3.
Summary of Dodoc benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	*R* Factor Low	*R* Factor High	Comments
Alliant FX/80; MP = 4	68020	—	32	—	—	248	248	64 bits
HP 9000 Mod. 370	68030	64	8-48	33.00	7.0	47	68	64 bits; L = 68882; H = Weitek
Alliant FX-1	Proprietary	—	—	12.00	—	85	85	64 bits
Alliant FX/8; MP = 8	Proprietary	—	—	—	35.0	101	101	
Amdahl 470 V8	—	—	—	—	—	150	150	64 bits
Amdahl 5860	—	—	—	—	—	475	475	64 bits
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	291	460	64 bits; H = Fortran V. 10.5r15
Bull DSP 90/x	—	—	—	—	—	371	371	64 bits
CCI Power 6/32	Proprietary	—	—	—	—	50	50	
CDC Cyber 990-E	Proprietary	—	—	—	6.2	592	592	64 bits
Celerity 1260	Proprietary	—	—	—	6.2	48	48	
Compaq 386/25	80386	32	5	25.00	—	73	73	64 bits; WTL3167 & DOS
Convex C-120	Proprietary	—	32	—	—	103	103	64 bits
Convex C-210; MP = 1	Proprietary	—	—	—	—	296	296	64 bits
Cray X-MP	Proprietary	—	—	—	—	1,080	1,080	
Cray X-MP/28	Proprietary	—	—	—	—	1,701	1,701	64 bits
Data Gen. MV20000	Proprietary	16	64	—	10.0	99	99	
Data Gen. MV40000	Proprietary	—	—	—	—	246	246	64 bits
DEC VAX 11/780 VMS	Proprietary	—	—	5.00	1.0	26	26	64 bits
DEC VAX 8600	Proprietary	—	—	—	4.5	91	91	VMS; 64 bits
DEC VAX 8650	Proprietary	—	—	—	6.2	129	129	
DEC VAX 8700	Proprietary	—	—	—	6.0	136	136	VMS; 64 bits
Decstation 3100	R2000	128	8	16.70	14.0	255	268	64 bits
Edge 1	Proprietary	—	—	—	—	53	53	
Floating-Pt. 264 SJE	Proprietary	—	—	—	—	396	396	64 bits
Fujitsu VP-200	—	—	—	—	—	915	915	64 bits
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	214	214	
HP 9000 Mod. 850S	Prop. RISC	—	—	13.70	7.0	201	201	
HP 9000 Mod. 855S	Prop. RISC	256	—	25.00	—	266	266	64 bits
Harris HCX-7	Proprietary	—	—	—	7.7	64	64	
IBM 3081G	—	—	—	—	—	181	181	
IBM 3081K	—	—	—	—	—	236	236	64 bits
IBM 3090	—	—	—	—	10.0	714	714	Scalar mode
IBM 3090/200	—	—	—	—	10.0	847	847	64 bits
IBM 4381 Mod. 2	Proprietary	—	—	—	—	90	90	64 bits
Intergraph Interpro C245	Clipper RISC	—	—	33.00	—	40	40	64 bits
Intergraph Interpro C370	Clipper RISC	—	—	40.00	—	72	72	64 bits
MIPS M/1000	R2000	128	16	15.00	15.0	227	227	
MIPS M/120-5	R2000	128	8	16.70	13.0	280	289	64 bits
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	438	443	64 bits
MIPS M/500	R2000	24	8	8.00	8.0	88	100	
MIPS M/800	R2000	128	8	12.50	12.5	190	19	
MIPS RC2030	R2000	64	16	16.70	12.0	238	238	64 bits
Silicon Graphics 4D/70	R2010	24	8	12.50	—	170	170	64 bits
Sun 3/110	68020	0	4	16.70	2.0	17	17	
Sun 3/160 W/FPA	68020	0	4	16.70	2.0	35	35	64 bits
Sun 3/260	68020	0	8-24	25.00	4.0	22	43	L = 68881; H = Weitek
Sun 3/260 w/FPA	68020	0	8-24	25.00	4.0	54	54	64 bits
Sun 386i Mod. 250	80386	32	16	25.00	—	26	26	64 bits
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	90	97	Weitek FPU; 64 bits
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	69	75	

Table 4.
Summary of Khornerstone benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Khornerstones/s Low	Khornerstones/s High	Comments
ALR FlexCache 25386	80386	64	5	25.00	—	3,917	3,917	Unix OS
Altos Series 2000	80386	—	—	16.00	3.0	3,038	3,038	Xenix
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	54,768	54,768	Up to four CPUs
Apollo Series DN3000	68020	—	—	12.50	1.2	2,469	2,469	
Apollo Series DN4000	68020	0	4	25.00	4.0	5,296	5,296	
Compaq 386	80386	0	4	16.00	—	1,941	1,941	
Compaq 386/20	80386	0	4	20.00	—	3,902	4,418	L = MS-DOS, H = Unix
Compaq 386/25	80386	32	5	25.00	—	5,037	7,417	L = SCO Unix
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	2,181	2,191	
Decstation 3100	R2000	128	8	16.70	14.0	5,206	5,206	
Dell System 325	80386	32	4	25.00	—	7,001	7,001	Unix
HP 9000 Mod. 340	68030	—	—	16.70	—	4,880	4,880	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	6,639	6,995	L = 68882; H = FPA
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	22,329	22,329	
IBM PS/2 Mod. 155SX	80386	0	2	16.00	—	2,041	2,041	
IBM PS/2 Mod. 70-121	80386	0	4	20.00	—	6,800	6,800	AIX
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	3,967	6,800	L = SCO Unix; H = AIX
IBM PS/2 Mod. 80	80386	—	—	16.00	2.8	2,265	2,265	
IBM RT PC	Prop. RISC	—	—	5.90	4.5	5,459	7,455	
IBM RT PC Mod. L135	Prop. RISC	—	—	7.30	6.0	7,296	7,296	W/FPA
MIPS M/120-5	R2000	128	8	16.70	13.0	18,819	23,218	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	26,177	26,177	
NCR Tower 32/400	68020	8	4	16.70	1.5	2,231	2,949	L = no cache; H = cache
NCR Tower 2/450	68020	8	4	25.00	—	3,932	3,932	
Solbourne 4/600	Sparc	64	16	16.70	7.0	14,515	14,515	
Sun 3/160C	68020	0	4	16.70	2.0	2,982	3,610	68881 FPU
Sun 3/260	68020	0	8-24	25.00	4.0	5,454	6,767	
Sun 3/50	68020	0	4	15.00	1.5	2,732	2,733	
Sun 3/60	68020	0	4	16.70	2.0	3,966	3,966	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	5,317	5,317	w/80387 FPU
Sun 386i Mod. 1250	80386	32	16	25.00	—	5,317	5,317	
Sun 4/260	Sparc	0	16-32	16.70	10.0	11,440	20,729	L = Weitek FPU
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	7,265	7,265	
Tektronix 4319	68020	0	4	20.00	2.5	4,114	4,114	

loads on a system and therefore does not accurately measure multiuser performance. (See Table 4.)

Linpack. This widely used benchmark provides a relative indication of system performance for engineering and scientific applications. Argonne National Laboratories wrote Linpack, which it also maintains. The routine tests vector performance on individual scientific applications while it stresses cache performance. As a linear-equations package, Linpack emphasizes floating-point addition and multiplication.[6] The results, measured in millions of floating-point operations per second (Mflops), are typically derived from a calculation of a 100 × 100 submatrix of linear equations. Since the benchmark is quite large, it does not completely fit into the cache space of most computers. Consequently, memory speed as well as floating-point processor performance can affect test results. Compilers can easily vectorize the routine, which results in higher performance ratings on those vector processor systems.

Linpack uses a set of general-purpose utilities called Basic Linear Algebra Subroutines (BLASs) to do the actual calculations. The benchmark subroutines can come in two forms: coded or Fortran. Some vendors use hand-coded, assembly-language versions of the library package, which typically yield improved benchmark performance results. The Fortran version comes with a set of standard algebraic subroutines (Fortran libraries). These Fortran BLASs come in two forms: unrolled and rolled. The body of the unrolled form contains an inner loop that is coded with multiple statements, while the rolled version contains a single statement that effectively performs the same operation. As would be expected, the rolled version of the Fortran BLAS yields higher (faster) benchmark ratings. (Table 5 reflects single-precision results.)

Livermore Fortran kernel. This benchmark provides insight into the system performance of scientific applications in both vector and nonvector processor environments. Lawrence Livermore National Laboratories developed the code. Since its work is dominated by large scientific calculations that can be vectorized, the laboratory developed this benchmark to evaluate large supercomputer systems. The benchmark has since migrated to the rest of the computer industry—even to personal computers.

The actual test consists of 24 sections of code taken from applications typically run by the laboratory. These kernels inhabit a larger benchmark driver that runs the routines several times, using different input data each time. The driver also checks on the accuracy and timing of the results and produces a statistical report of the test.[2] The results appear as 24 separate numbers (one for each kernel) as Mflop measurements for three different vector lengths, which total 72 results. Various statistical means (arithmetic, geometric, harmonic) provide insight into general-system performance. An analysis performed by Livermore Labs suggests that each statistical mean corresponds to a level of system vectorization.[2] In terms of vectorization, the harmonic, geometric, and arithmetic means approximate 40, 70, and 90 percent. These three results are often interpreted as

Table 5.
Summary of single-precision Linpack benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Mflops Low	Mflops High	Comments
Altos Series 2000	80386	—	—	16.00	3.0	0.11	0.11	
Apollo DN660	Proprietary	—	—	—	1.0	0.1	0.1	
Apollo Series DN3000	68020	—	—	12.50	1.2	0.07	0.073	
Apollo Series DN4000	68020	0	4	25.00	4.0	0.14	0.14	
Celerity 1200	Proprietary	—	—	—	2.3	300.0	300.0	
Compaq 386/20 w/80387	80386	0	4	20.00	—	0.14	0.26	
Compaq 386/20 w/Weitek	80386	0	4	20.00	—	0.64	0.64	
Data Gen. MV10000	Proprietary	—	—	—	2.5	390.0	390.0	
DEC VAX 11/785	Proprietary	—	—	7.50	1.5	0.23	0.23	w/FPA
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	0.162	0.162	
HP 9000 Mod. 340	68030	—	—	16.70	—	0.167	0.168	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	0.24	0.66	L = 68882; H = FPA
HP 9000 Mod. 835SRX	Prop. RISC	128	—	15.00	4.0	2.29	2.29	
IBM PC AT	80286	—	—	—	—	0.013	0.013	
IBM RT PC	Prop. RISC	—	—	5.90	4.5	0.11	0.36	L = 68881; H = FPA
IBM RT PC Mod. 135	Prop. RISC	—	—	7.30	6.0	0.44	0.44	w/FPA
Motorola MVME181-2	88100	32	—	25.00	—	1.8	1.82	
Motorola MVME188SP-1	88100	128	16	20.00	17.0	1.71	1.77	
Motorola SYS8800	88100	128	16	20.00	17.0	1.71	1.77	
NCR Tower 32/400	68020	8	4	16.70	1.5	0.095	0.1	L = no cache; H = cache
NCR Tower 32/450	68020	8	4	25.00	—	0.217	0.217	
Pyramid 90x/FP	Proprietary	—	—	8.00	2.5	200.0	200.0	
Sun 3/50	68020	0	4	15.00	1.5	0.092	0.092	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	0.25	0.25	w/80387 FPU
Tektronix 4319	68020	0	4	20.00	2.5	0.105	0.105	

Table 6.
Summary of single-precision Livermore Fortran kernel benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Mflops Low	Mflops High	Comments
Acer 1100/25	80386	32	4	25.00	—	0.097	0.12	L = DOS/X; H = Unix
Alliant FX-1, Scalar	Proprietary	—	—	12.00	—	0.66	0.66	
Alliant FX-1, Vector	Proprietary	—	—	12.00	—	0.6	0.6	
Alliant FX/8; MP = 8	Proprietary	—	—	—	35.0	1.3	1.31	
ALR FlexCache 25386	80386	64	5	25.00	—	0.08	0.12	L = DOS/X; H = Unix
AMI Mark II	80386	64	—	33.00		0.3	0.3	w/Weitek 3167
AST 386/33	80386	0	—	33.00	—	0.13	0.275	L = 80387; H = Weitek 3167
AST 386C-390	80386	64	4	25.00	—	0.063	0.08	L = DOS/X; H =Unix
Cheetah cAT 386	80386	0	5	20.00	—	0.05	0.05	
Compaq 386/25 w/80387	80386	32	5	25.00	—	0.09	0.133	L = DOS/X; H = DOS/X
Compaq 386/25	80386	32	5	25.00	—	0.22	0.22	w/Weitek
Convex C-1 Scalar	Proprietary	—	—	—	4.5	1.11	1.11	
Convex C-1 Vector	Proprietary	—	—	—	4.5	1.27	1.27	
DEC VAX 11/780 4.3 BSD	Proprietary	—	—	5.00	1.0	0.18	0.18	
DEC VAX 11/780 VMS	Proprietary	—	—	5.00	1.0	0.28	0.28	
DEC VAX 8700 VMS	Proprietary	—	—	—	6.0	1.26	1.26	
Dell System 310	80386	32	3	20.00	—	0.07	0.09	L = DOS/X; H = Unix
Dell System 325	80386	32	4	25.00	—	0.1	0.12	L = DOS/X; H = Unix
Elxsi 6420	Proprietary	—	16	20.00	—	1.31	1.31	
Everex Step 386/25	80386	64	4	25.00	—	0.09	0.11	L = DOS/X; H = Unix
Fivestar 386/20	80386	64	4	20.00	—	0.073	0.091	L = DOS/X; H = Unix
HP Vectra RS/25C	80386	32	4	25.00	—	0.088	0.109	L = DOS/X; H = DOS/X
Hertz 386/25	80386	64	4	25.00	—	0.1	0.11	L = DOS/X; H = Unix
IBM PS/2 Mod. 70-121	80386	0	4	20.00	—	0.06	0.08	L = DOS/X; H = Unix
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	0.09	0.11	L = DOS/X; H = Unix
Micro Express 386/25	80386	64	4	25.00	—	0.101	0.121	L = DOS/X; H = DOS/X
MIPS M/1000	R2000	128	16	15.00	15.0	2.02	2.29	
MIPS M/120-5	R2000	128	8	16.70	13.0	2.58	2.85	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	3.8	4.24	
MIPS M/500	R2000	24	8	8.00	8.0	0.97	0.97	
MIPS RC2030	R2000	64	16	16.70	12.0	2.82	2.82	
Proteus 4400GL	80386	64	4	25.00	—	0.101	0.121	L = DOS/X; H = DOS/X
Rupp 386/20	80386	0	5	20.00	—	0.056	0.056	
Sun 3/160	68020	0	4	16.70	2.0	0.46	0.46	w/FPA
Sun 3/260	68020	0	8-24	25.00	4.0	0.65	1.04	
Sun 3/60	68020	0	4	16.70	2.0	0.14	0.14	
Sun 386i Mod. 250	80386	32	16	25.00	—	0.08	0.08	
Sun 4/260	Sparc	0	16-32	16.70	10.0	1.03	1.04	w/Weitek
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	0.77	0.77	
Tandon 386/20	80386	64	4	20.00	—	0.08	0.09	L = DOS/X; H = Unix
Tandy 4000 LX	80386	0	16	20.00	—	0.07	0.08	L = DOS/X; H = Unix

separate benchmark tests. (Table 6 reflects single-precision results at 40-percent vectorization.)

SPICE. The general-purpose Simulation Program with Integrated Circuit Emphasis came from the University of California at Berkeley. This benchmark makes heavy use of both integer and double-precision, floating-point calculations (the floating-point operations are not vector oriented). Because it is quite large, the program is a good test of system instruction and data-cache performance. SPICE accepts a circuit description as input and simulates the design. The user can monitor currents and voltages at various circuit locations.

Table 7.
Summary of SPICE benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Seconds Low	Seconds High	Comments
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	5.0	5.1	Up to four CPUs
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	154.4	154.4	w/FPA; Unix 4.2 BSD
DEC VAX 8600	Proprietary	—	—	—	4.5	28.0	28.0	Unix
Decstation 3100	R2000	128	8	16.70	14.0	32.13	32.13	
Sun 3/260	68020	0	8-24	25.00	4.0	31.0	31.0	
Sun 4/260	Sparc	0	16-32	16.70	10.0	19.0	19.0	
Sun 4/260	Sparc	0	16-32	16.70	10.0	61.78	61.78	w/Weitek
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	79.87	79.87	

UC Berkeley and several system vendors have distributed various input data packs for simulation of different types of circuits. The user must take care to ensure that benchmark results from different systems have used the same circuit simulations. The number of seconds required to perform the simulation constitutes the test-result measurements. (Table 7 reflects data from 2G6 circuit simulations only.)

Stanford Integer. John Hennessey of Stanford University compiled this benchmark test. Written in the C programming language, the suite consists of a series of small programs that use algorithms to solve real-world problems.

Some of these small programs include the Towers of Hanoi and the Eight Queens puzzles, multiplication of integer matrices, and the quick and bubble sorts. Yet another routine inserts and recursively searches a binary tree. Test measurements consist of the geometric mean of all results displayed in either seconds or milliseconds (See Table 8.)

Stanford Floating Point. The program uses tight loops and a large proportion of floating-point code to calculate the results.[2] The routines' susceptibility to code optimization by high-quality compilers affects the results. The suite consists of the fast Fourier transform (FFT) and matrix multiplication (MM) tests. The first test typically computes a 256-point, single-precision FFT 20 times. The second test multiplies two 40 × 40 single-precision matrices. The results appear in either seconds or milliseconds. (See Table 8.)

Table 8.
Summary of Stanford benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Milliseconds Low	Milliseconds High	Comments
Stanford Integer								
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	1,550	2,140	
DEC VAX 8550	Proprietary	—	—	—	6.4	260	350	
DEC VAX 8600	Proprietary	—	—	—	4.5	620	860	
DEC VAX 8810	Proprietary	—	—	22.22	12.0	6.5	6.5	
Decstation 3100	R2000	128	8	16.70	14.0	115	115	
MIPS M/1000	R2000	128	16	15.00	15.0	130	130	
MIPS M/120-5	R2000	128	8	16.70	13.0	112	118	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	67	75	
MIPS M/500	R2000	24	8	8.00	8.0	260	260	
MIPS M/800	R2000	128	8	15.00	15.0	160	160	
MIPS RC2030	R2000	64	16	16.70	12.0	110	110	
Sun 3/160	68020	0	4	16.70	2.0	530	530	

Table 8.
Summary of Stanford benchmark test results (continued).

System/model	Processor	Cache	RAM	Frequency	MIPS	Milliseconds		Comments
		(Kbytes)	(Mbytes)	(MHz)	rating	Low	High	
Sun 3/260	68020	0	8-24	25.00	4.0	360	360	
Sun 4/260	Sparc	0	16-32	16.70	10.0	150		150 w/Weitek
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	222	222	
Stanford Floating Point								
DEC µVAX II w/Ultrix	Proprietary	—	—	5.00	0.9	3,000	3,000	Single-precision MM
DEC µVAX II w/Ultrix	Proprietary	—	—	5.00	0.9	4,900	4,900	256-point FFT
DEC µVAX II w/VMS	Proprietary	—	—	5.00	0.9	1,900	1,900	Single-precision MM
DEC µVAX II w/VMS	Proprietary	—	—	5.00	0.9	3,300	3,300	256-point FFT
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	2,040	2,310	Single-precision MM
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	3,870	3,970	256-point FFT
DEC VAX 8600	Proprietary	—	—	—	4.5	1,370	1,370	Single-precision MM
DEC VAX 8600	Proprietary	—	—	—	4.5	1,420	1,420	256-point FFT
MIPS M/1000	R2000	128	16	15.00	15.0	120	120	Single-precision MM
MIPS M/1000	R2000	128	16	15.00	15.0	170	170	256-point FFT
MIPS M/120-5	R2000	128	8	16.70	13.0	54	70	Single-precision MM
MIPS M/120-5	R2000	128	8	16.70	13.0	95	100	256-point FFT
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	35	43	Single-precision MM
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	64	64	256-point FFT
MIPS M/500	R2000	24	8	8.00	8.0	260	260	Single-precision MM
MIPS M/500	R2000	24	8	8.00	8.0	340	340	256-point FFT
MIPS M/800	R2000	128	8	15.00	15.0	120	130	Single-precision MM
MIPS M/800	R2000	128	8	15.00	15.0	170	200	256-point FFT
MIPS RC2030	R2000	64	16	16.70	12.0	55	55	Single-precision MM
MIPS RC2030	R2000	64	16	16.70	12.0	98	98	256-point FFT
Sun 3/160	68020	0	4	16.70	2.0	498	500	Single-precision MM
Sun 3/160	68020	0	4	16.70	2.0	840	840	256-point FFT
Sun 3/260	68020	0	8-24	25.00	4.0	250	380	Single-precision MM
Sun 3/260	68020	0	8-24	25.00	4.0	460	600	256-point FFT
Sun 3/60	68020	0	4	16.70	2.0	870	870	Single-precision MM
Sun 3/60	68020	0	4	16.70	2.0	1,550	1,550	256-point FFT
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	250	263	Single-precision MM
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	460	560	256-point FFT

Whetstone. This benchmark is a synthetic mix of integer and floating-point calculations, transcendental functions, conditional jumps, function calls, and array indexing. The code usually comes in a Fortran version. Considered a classic, the benchmark was designed to represent typical scientific programs.

The original version emerged from an analysis of 949 Algol-60 programs.[7] The structure of Whetstone defies vectorization by many optimizing compilers. Results display in thousands or millions of Whetstone interpreter instructions per second (KWhips or MWhips, sometimes referred to as MegaWhetstones/s).

This benchmark has fallen out of favor with the computer industry in recent years. So many different versions of Whetstone (written in either Fortran or the C programming language) make correlating the data derived from two different sources almost impossible. (Table 9 reflects single-precision results.)

Table 9.
Summary of single-precision Whetstone benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	MWhips Low	MWhips High	Comments
Acer 1100/25	80386	32	4	25.00	—	2.05	2.42	L = Unix; H = DOS/X
Alliant FX/8; MP = 8	Proprietary	—	—	—	35.0	4.9	4.9	
ALR FlexCache 25386	80386	64	5	25.00	—	2.03	2.098	
Altos Series 2000	80386	—	—	16.00	3.0	1.18	1.18	
AMI Mark II	80386	64	—	33.00	—	7.11	7.11	w/Weitek 3167
Apollo 5X0T	68020	—	—	20.00	3.4	7.74	2.2	L = no FPU; H = FPU
Apollo DN660	Proprietary	—	—	—	1.0	0.69	0.69	
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	16.97	16.97	Up to four CPUs
Apollo Series DN3000	68020	—	—	12.50	1.2	0.716	0.78	
Apollo Series DN4000	68020	0	4	25.00	4.0	1.89	2.17	
Apollo Series DN4500	68030	64	16	33.00	7.0	3.23	3.23	
Apple Macintosh II	68020	—	—	15.70	—	0.52	0.05	
Apple Macintosh IIx	68030	0	4	16.70	—	0.73	0.73	
Apple Macintosh IIxc	68030	0	4	16.70	—	0.73	0.73	
Apple Macintosh SE	68020	0	—	—	—	0.05	0.05	
Apple Macintosh SE/30	68030	0	4	16.70	—	0.73	0.73	
AST 386/33	80386	0	—	33.00	—	3.66	7.05	L = 80387; H = Weitek 3167
AST 386C-390	80386	64	4	25.00	—	3.66	7.05	L = OS/2; H = DOS/X
CCI Power 7/64	Proprietary	—	—	—	—	14.07	14.07	
Celerity 1200	Proprietary	—	—	—	2.3	0.23	0.23	
Celerity 1260	Proprietary	—	—	—	6.2	0.88	0.88	
Cheetah cAT 386	80386	0	5	20.00	—	1.11	1.53	L = Unix; H = DOS/X
Compaq 386/20 w/80387	80386	0	4	20.00	—	1.72	1.89	
Compaq 386/20 w/Weitek	80386	0	4	20.00	—	4.12	4.31	
Compaq 386/25 w/80387	80386	32	5	25.00	—	2.0	2.4	L = Unix; H = DOS/X
Compaq 386/25 w/Weitek	80386	32	5	25.00	—	5.22	5.22	
Data Gen. MV10000	Proprietary	—	—	—	2.5	2.8	2.8	
Data Gen. MV1400 DC	Proprietary	0	12	6.25	—	0.97	0.97	
Data Gen. MV15000-10	Proprietary	16	64	11.80	4.0	4.6	4.6	
Data Gen. MV15000-20	Proprietary	16	64	11.80	8.0	7.13	7.13	
Data Gen. MV15000-8	Proprietary	16	64	11.80	4.0	3.06	3.06	
Data Gen. MV2000 DC	Proprietary	0	12	6.25	—	0.9	0.97	
Data Gen. MV20000	Proprietary	16	64	—	10.0	5.9	5.9	
Data Gen. MV2500 DC	Proprietary	0	24	20.00	—	1.62	1.62	
Data Gen. MV7800	Proprietary	0	14	3.13	—	1.13	1.13	
Data Gen. MV7800 DC	Proprietary	0	14	3.13	—	1.13	1.13	
Data Gen. MV7800 DCX	Proprietary	0	14	4.50	—	1.58	1.58	
Data Gen. MV7800 XP	Proprietary	0	14	4.50	—	1.58	1.58	
DEC μVAX 3500 w/VMS	Proprietary	—	—	11.10	3.5	0.47	0.47	
DEC μVAX II w/Ultrix	Proprietary	—	—	5.00	0.9	0.4	0.4	
DEC μVAX II w/VMS	Proprietary	—	—	5.00	0.9	0.93	0.93	
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	1.313	1.313	
DEC VAX 11/780 4.3 BSD	Proprietary	—	—	5.00	1.0	0.5	1.08	
DEC VAX 11/785	Proprietary	—	—	7.50	1.5	1.8	1.8	w/FPA
DEC VAX 8600	Proprietary	—	—	—	4.5	4.6	4.6	
DEC VAX 8650	Proprietary	—	—	—	6.2	6.1	6.9	
DEC VAX 8700	Proprietary	—	—	—	6.0	5.9	6.67	
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	0.47	0.47	
Decstation 3100	R2000	128	8	16.70	14.0	11.5	13.0	
Definicon Sparc 1	Sparc	—	4	20.00	—	4.4	4.4	
Dell System 310	80386	32	3	20.00	—	1.6	2.0	L = Unix; H = DOS/X
Dell System 325	80386	32	4	25.00	—	2.0	2.4	L = Unix; H = DOS/X
Everex Step 386/25	80386	64	4	25.00	—	2.0	2.4	L = Unix; H = DOS/X
Fivestar 386/20	80386	64	4	20.00	—	1.54	1.83	L = Unix; H = DOS/X
HP 9000 Mod. 340	68030	—	—	16.70	—	1.713	1.73	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	2.1	3.0	L = 68882; H = FPA

Table 9.
Summary of single-precision Whetstone benchmark test results (continued).

		Cache	RAM	Frequency	MIPS	MWhips		
System/model	Processor	(Kbytes)	(Mbytes)	(MHz)	rating	Low	High	Comments
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	2.1	3.0	L = 68882; H = FPA
HP 9000 Mod. 370	68030	64	8-48	33.00	7.0	3.41	3.41	
HP 9000 Mod. 825S	Prop. RISC	16	—	12.50	3.0	3.5	3.58	
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	9.0	9.33	
HP 9000 Mod. 840S	Prop. RISC	128	24	8.00	4.5	3.1	3.1	
HP 9000 Mod. 850S	Prop. RISC	—	—	13.70	7.0	4.2	8.1	
HP Vectra RS/25C	80386	32	4	25.00	—	1.95	2.3	L = Unix; H = DOS/X
Harris HCX-7	Proprietary	—	—	—	7.7	7.1	7.1	
Hertz 386/25	80386	64	4	25.00	—	2.0	2.3	L = Unix; H = DOS/X
IBM 3081D	—	—	—	—	—	5.85	5.85	
IBM 3090	—	—	—	—	10.0	18.0	18.0	
IBM PS/2 Mod. 70-121	80386	0	4	20.00	—	1.35	1.63	L = Unix; H = DOS/X
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	1.94	2.538	L = Unix; H = AIX
IBM RT PC	Prop. RISC	—	—	5.90	4.5	0.81	1.64	L = 68881; H = FPA
IBM RT PC Mod. 135	Prop. RISC	—	—	7.30	6.0	2.12	2.12	w/FPA
Integr. Sol. Advantage2000	R2000	64	16	16.70	12.0	11.4	11.4	
Intergraph Interpro 32C	Clipper RISC	—	—	30.00	5.0	2.98	2.98	
Micro Express 386/25	80386	64	4	25.00	—	2.03	2.44	L = Unix; H = DOS/X
MIPS M/1000	R2000	128	16	15.00	15.0	10.28	10.5	
MIPS M/120-3	R2000	128	8	12.50	10.0	9.1	9.1	
MIPS M/120-5	R2000	128	8	16.70	13.0	11.4	12.1	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	13.8	18.0	L = 20 MHz; H = 25 MHz
MIPS M/500	R2000	24	8	8.00	8.0	5.43	5.43	R2360 FPU
MIPS M/800	R2000	128	8	15.00	15.0	8.57	8.57	
MIPS RC2030	R2000	64	16	16.70	12.0	12.1	12.1	
Motorola MVME141-1	68030	64	4	25.00	5.3	0.69	1.88	
Motorola MVME141-2	68030	64	4	33.00	6.9	2.5	2.5	
Motorola MVME147	68030	0	4	20.00	3.8	1.43	1.51	
Motorola MVME147-1	68030	0	4	25.00	4.7	1.79	1.96	
Motorola MVME180	88100	32	—	20.00	17.0	12.7	18.2	
Motorola MVME181-2	88100	32	—	25.00	—	15.9	22.8	
Motorola SYS1147	68030	0	4	20.00	3.8	1.43	1.43	
Motorola SYS2300	68020	0	4	16.70	1.5	0.93	0.93	
Motorola SYS2600	68020	16	4	16.70	1.5	0.87	0.87	
Motorola SYS3300	68030	0	4	20.00	3.8	1.43	1.43	
Motorola SYS3600	68030	0	4	25.00	4.7	1.79	1.79	
Motorola SYS3640	68030	64	4	25.00	5.3	1.79	1.79	
NCR Tower 32/400	68020	8	4	16.70	1.5	0.864	0.933	L = no cache; H = cache
NCR Tower 32/450	68020	8	4	25.00	—	1.65	1.65	
Opus Systems	88000	32	4-20	20.00	17.0	18.58	18.58	
Opus Systems	88000	32	4-20	20.00	17.0	9.43	9.43	
Prime EXL 316	80386	—	—	16.00	3.2	3.3	3.3	
Proteus 4400GL	80386	64	4	25.00	—	2.03	2.44	L = Unix; H = DOS/X
Rupp 386/20	80386	0	5	20.00	—	1.11	1.53	L = Unix; H = DOS/X
Silicon Graphics Iris	R2010	24	8	12.50	—	7.35	7.35	
Solbourne 4/600	Sparc	64	16	16.70	7.0	6.65	6.65	
Sun 3/160C w/68881	68020	0	4	16.70	2.0	1.03	2.6	L = 68881; H = FPA
Sun 3/260	68020	0	8-24	25.00	4.0	1.25	5.3	L = 68881; H = FPA
Sun 3/50	68020	0	4	15.00	1.5	0.71	0.936	w/68881
Sun 3/60	68020	0	4	16.70	2.0	1.27	1.42	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	1.92	1.92	w/80387 FPU
Sun 386i Mod. 250	80386	32	16	25.00	—	1.63	1.923	
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	4.6	5.66	Weitek FPU
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	4.22	4.28	
Tandon 386/20	80386	64	4	20.00	—	1.5	1.9	L = Unix; H = DOS/X
Tandy 4000 LX	80386	0	16	20.00	—	1.43	1.71	L = Unix; H = DOS/X

The list of public-domain, licensed third-party, and proprietary benchmark tests in use today seems endless. This plethora, combined with the various benchmark revisions, testing procedures, and analysis methods, complicates the task of comparing general system performance. Understanding simple concepts like what a benchmark does—and how the results are produced—provides a valuable first step in the right direction.

Acknowledgment

I especially thank Tom Ritch of the Motorola Microcomputer Division for editorial and technical assistance during the development of this article.

References

1. Jim Geers, "A New Generation of Benchmarking," *Mips*, Vol. 1, No. 1, Feb. 1989, pp. 92-100.
2. *Performance Brief—CPU Benchmarks*, Issue 3.5, Mips Computer Systems, Sunnyvale, Calif., Oct. 1988.
3. *Am29000 Performance Analysis*, Advanced Micro Devices, Sunnyvale, Calif., May 1988.
4. David Bouffard et al., *DECstation 3100 Performance Summary*, Version 1.0, Digital Equipment Corporation, Maynard, Mass., Jan. 10, 1989.
5. David Wilson, "Tested Mettle," *UNIX Review*, Volume 7, No. 1, Jan. 1989, pp. 97-107.
6. J.J. Dongarram, *Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment*, Argonne National Laboratory, Argonne, Ill., Feb. 16, 1988.
7. Bill Nicholls, "The 'B' Word!," *Byte*, June 1988, pp. 207-212.

Walter J. Price is a senior marketing analyst with the Microcomputer Division of Motorola, where he performs competitive and market analyses of board- and system-level products. His current interests include collecting comprehensive information regarding the microcomputer industry.

Price received the BS degree in electronics technology and computer science from Northern Michigan University in Marquette.

Questions about this article may be directed to the author at Motorola, Microcomputer Division, 2900 South Diablo Way, Tempe, AZ 85282.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 165 **Medium** 166 **High** 167

Hardware Monitoring of a Multiprocessor System

This device records events in a time mode for straightforward reconstruction and supports fault tolerance in distributed systems.

An-Chi Liu
Feng Chia University

Ranjani Parthasarathi
Illinois Institute of Technology

Advances in semiconductor and very large scale integration technology have greatly decreased the cost of computer hardware. This factor has boosted research in areas that range from high-performance computing to distributed processing. As computer systems become correspondingly more complex, performance evaluation also becomes more essential—and increasingly complicated.

Performance studies serve three purposes: the design, selection, and improvement of systems. For these studies we use tools that either model or monitor the target system. Modeling—whether it is analytical or simulated—occurs at the design stage. Monitoring, on the other hand, aids the study of the dynamic behavior of a system. A monitor observes the system during its normal operation and collects runtime data. Analysis of this data provides accurate performance information for system analysis, verification, and management. Performance management, for example, ensures that preassigned specifications are met and maintained at an acceptable level of efficiency.

In a multiprocessor environment, one must evaluate processing elements individually and collectively. Modeling a multiprocessor environment is an extremely complex task. It is much more effective to design a proper monitor to measure performance.

Terms

One can categorize monitors in terms of instrumentational, functional, or architectural issues.

Software vs. hardware. Software monitors consist of programs that detect the states of a system. These monitors also comprise sets of instructions called software probes that detect events.[1]

Hardware monitors, on the other hand, are electronic devices connected to specific system points via probes. These probes detect signals that characterize the phenomena to be observed. While software monitors function at a logical level, their hardware counterparts work at a physical level.

The interference in a target system caused by the introduction of the monitoring device is called an artifact.[2] To ensure an unbiased indication of performance, one should ascertain an artifact's magnitude and remove it from the measurements.

0272-1732/89/1000-0044$01.00 © 1989 IEEE

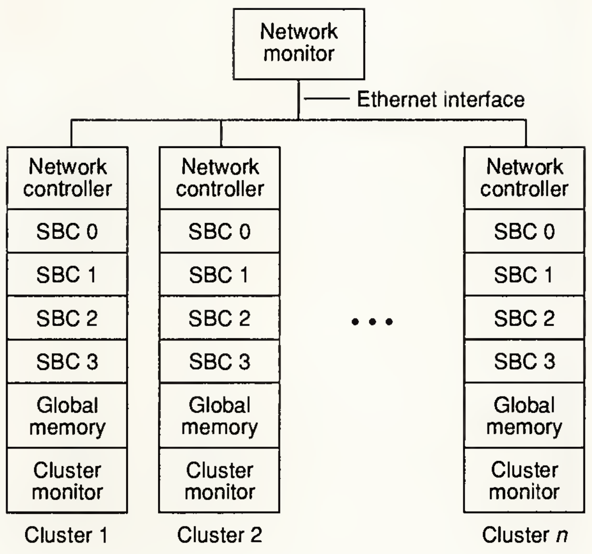


Figure 1. The Ted architecture.

Passive vs. active. The basic function of a monitor is passive in that it observes, collects, stores, and outputs data to the external world. An active monitor, however, further analyzes the data and takes corrective action based on this analysis. It dynamically controls the operation of the entire system to maintain performance specifications.

Centralized vs. distributed. A centralized monitor taps into critical points that provide primary information about the system. These points typically serve as the communication data paths. However, the selection of a location in the system that is central to all processing activity is a design issue. Since a centralized location is not generally feasible, our device collects only a selected set of data.

A distributed monitor associates multiple monitors with corresponding processors. Each monitor periodically sends the data it captures to a central analyzer. In a master-slave environment, the central analyzer can function as the master monitor that controls the functioning of the slave monitors. The slave monitors are totally passive and communicate with the master by means of a dedicated path.

Here we describe the design and implementation of a hardware monitor that observes an experimental multiprocessor system at certain critical points and selectively gathers data required for performance analysis.

The target environment

The Testbed for Distributed Processing, or Ted, consists of Intel Corp.'s iSBC 8086 Single Board Computers (SBCs) organized into groups or clusters. (Figure 1 summarizes the environment.) Each cluster consists of several SBCs that communicate via a shared memory. Intercluster communication occurs through an Ethernet interface. Both loosely coupled and tightly coupled environments exist in this architecture. Within a cluster, processors are tightly coupled and processes are shared between processors. Across clusters, processors are loosely coupled.

Developing the distributed kernel for Ted required support for process migration. Processes must migrate from one cluster to another and be allocatable to any processor in the system. The decision of when and how to export a process or a group of processes rests on factors such as processor workload and node capability. A monitor that collects the dynamic system state and passes it onto the system scheduler furnishes this information. A monitor is also necessary for debugging a distributed system.

The monitoring of Ted concerns both intercluster and intracluster traffic. Intercluster traffic is synonymous with Ethernet traffic. Since an Ethernet controller board provides the interface, traffic data on Ethernet can be obtained from the controller or from tools such as packet analyzers. Our focus with Ted is therefore intracluster monitoring.

Separate monitoring of each cluster implies that the monitor is distributed. In our implementation, each cluster contains one monitor that sits on its respective Multibus and overlooks the bus transactions. The monitor can be programmed to look for a particular set of events that trigger an associated action on the hardware monitor.

The monitor gathers data that can either be stored for future analysis or used for real-time dynamic control of the system. The device can perform either passive or active monitoring. The present monitor is implemented in a passive manner. An external system performs the analysis of the gathered data. For simplicity, we can use any one of the SBCs in the cluster as the external system. We plan to eventually achieve active monitoring to fine-tune Ted's performance.

Design philosophy

Figure 2 on the next page demonstrates the basic building blocks of a typical hardware monitor. Probes detect significant events. An event filter logically processes and selects these signals. The processing element normally comprises counters that obtain timing information. Collected data is recorded on storage media in the form of a secondary storage device or RAM buffer. The control unit coordinates the entire operation.

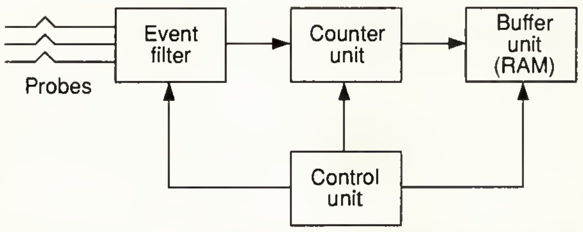


Figure 2. Functional block diagram of a hardware monitor.

The general design considerations for the Ted monitor include data reduction, programmability, communication paths, and testability.

Data reduction. Because the monitor observes every cycle of the bus, we could obtain a trace or profile of the Multibus (shown later in Figure 3) transactions. However, such data collection would demand excessive storage capacity. To reduce the amount of generated data, the monitor records only specified events or signals on the bus.

The monitor employs two basic approaches. It either counts the number of times a combination of events occurs, or it keeps track of the elapsed time between two events in terms of the number of clock cycles. Data collected in the count mode represents only an average value of the monitored event. In the time mode, the collected data helps reconstruct the sequence of events. We chose the time mode for this project.

Programmability. A flexible monitor that hunts for different signal patterns must include a user interface that allows a template of events as instructions. For example, one can program several consecutive Multibus addresses in shared memory into a kernel to indicate process states in a multiprocessor. In this case, we monitored a range of memory addresses. The user interface supports a signal pattern, a "don't-care" pattern (to select a range of memory addresses), and time duration.

Communication paths. The monitor communicates with the external devices via special data paths. These paths exchange the input instructions as well as collected data. This amount of activity often loads the bus. Therefore, it is essential that data-path activity does not interfere with the system being monitored.

An SBC could send instructions for monitor programming via the Multibus or a dedicated path. These messages take no more than a few bus cycles and can be removed in later calculations. Data communications from the monitor to the SBC, however, demand more bus usage. These communications occur via a dedicated path to avoid bus contention.

Therefore we used two separate simplex paths: SBC-to-monitor communications use the Multibus, while monitor-to-SBC messages use the dedicated path.

Testability. Testability—an essential part of any hardware design—is imperative to the determination of proper monitor functionality. If the activity on the bus is known, we can check the collected data for its validity. But when multiple-processor bus contention occurs, we find it difficult to keep track of—or control—bus traffic. Hence we used a method of testing the monitor's functionality without the presence of any external input.

The Ted monitor

Figure 3 is a block diagram of the Ted monitor.

Probes. The probes of the monitor connect to the bus within the cluster. A standard Multibus edge connector serves this function. (Note that we had to implement an interface to the bus.[3] Also, we used a buffered receiver for all signals running from the bus to the monitor to further reduce the loading.) The bus-priority logic on the motherboard helps derive the signals that represent which SBC has access to the bus. One interrupt line of the bus is reserved for monitor usage.

Event filter. An event filter looks for a set of events specified in terms of the signals that are being monitored. This filter can be programmed as a bit pattern for comparison with incoming signals. This unit consists of a set of registers (as shown in Figure 4) to store the bit pattern that represents the event, as well as a comparator to match the signals on the bus. The bit pattern consists of the following signal lines:

- 20-bit address lines,
- 16-bit data lines, and
- 4-bit read/write lines (I/O and memory read/write).

Thus we implemented a pattern register (PR) of 40 bits. A comparator unit matches all or part of the signals that appear on the bus. These signals are latched in the bus register. We employed a don't-care mask register to specify which bits are to be compared, using bit-wise OR logic. Both the pattern and the mask registers determine these bits. The output of a comparator activates a set of counters.

Counter unit. This unit implements the time mode of operation. The timing diagram in Figure 5 illustrates the mechanics of the counter unit. The signals match the pattern during clock cycles 1, 3, 4, 7, 8, and 9. The output signal Equal is high for the clock cycles that have an active pattern on the bus and is low for the rest. The period of observation totals 12 clock cycles; the

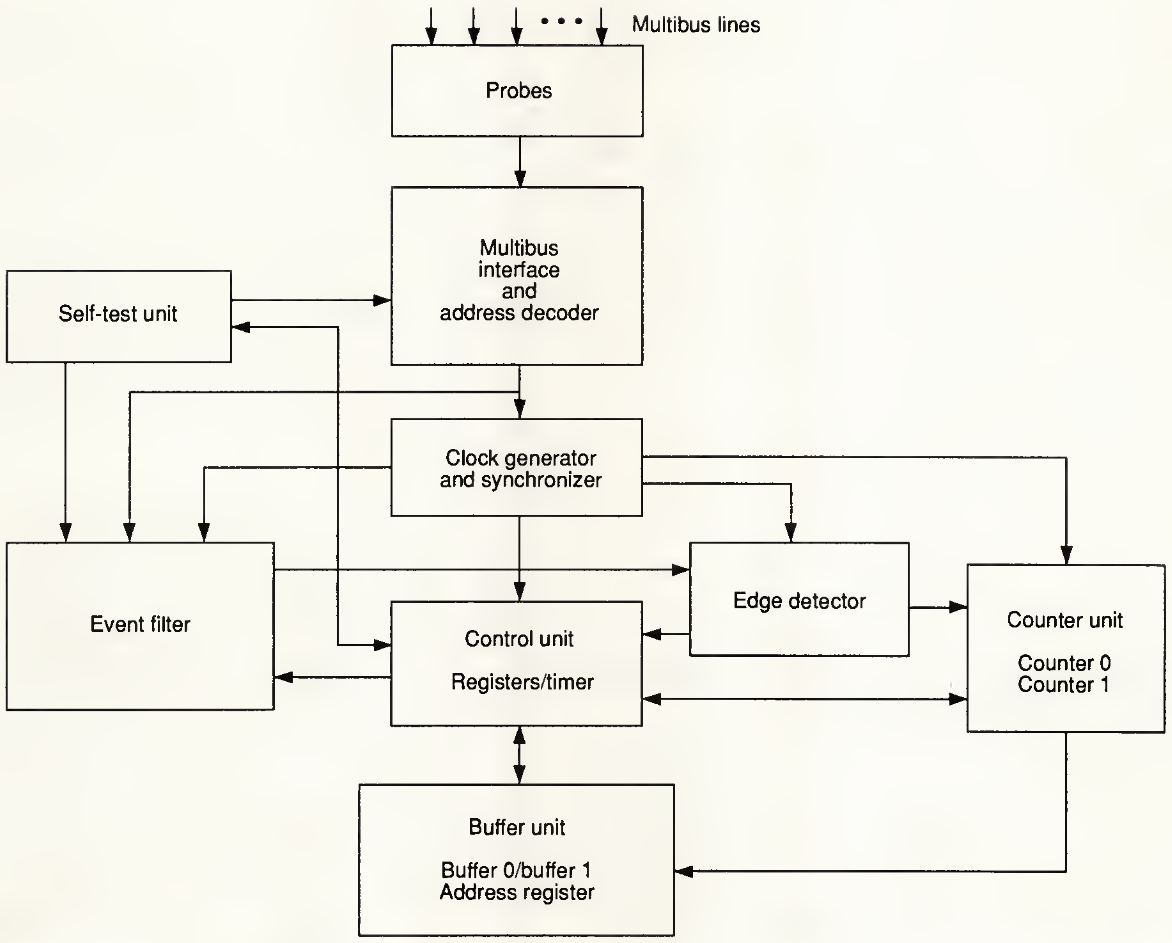


Figure 3. Ted hardware monitor.

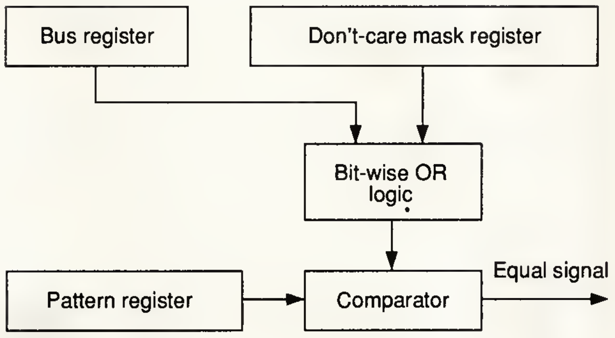


Figure 4. Registers in the event filter.

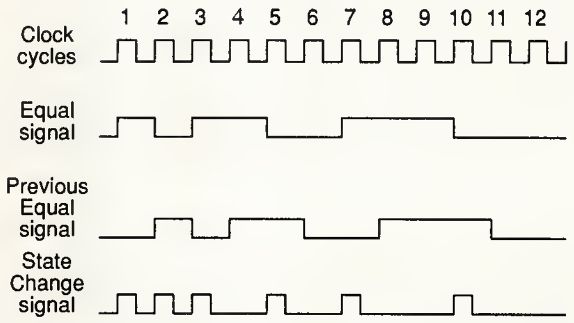


Figure 5. Timing diagram of the time-mode operation.

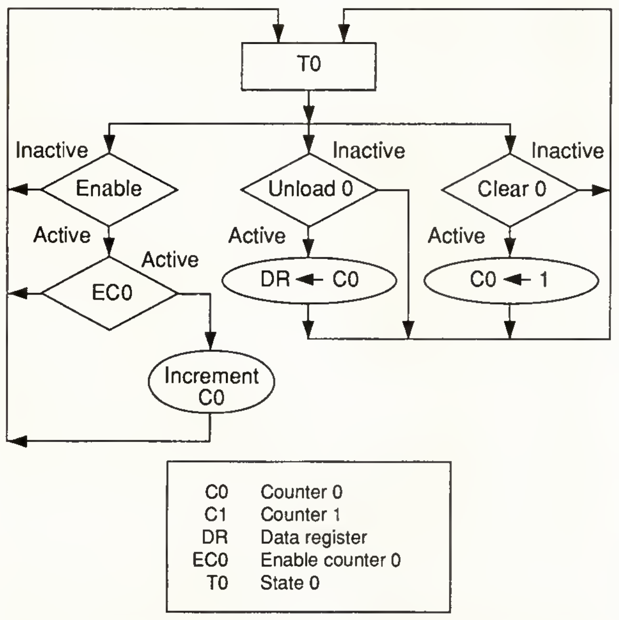


Figure 6. ASM chart for counter 0.

pattern becomes active and idle in spurts. The time mode requires that the pattern be recorded as alternative counts of active and idle times. For this example, the pattern is active at clock cycle 1, inactive at clock cycle 2, active for two cycles starting at clock cycle 3, inactive for two cycles starting at clock cycle 5, and so on, as indicated by the Equal signal in the diagram.

One counter measures the amount of inactive time and the other measures active time. The control unit enables the counters with the help of the State Change signal, which is derived by an exclusive OR of the present and previous Equal signals. Using two counters overlaps the counting and recording processes. Once a state change is detected, the next counter activates. At the same time, the system unloads and clears the first counter and keeps it ready for the next state change.

The system reserves one bit of the data to ascertain whether it pertains to active or inactive time. The monitor controls the two counters as a synchronous pair. Figure 6 shows the control for the inactive counter 0 in the form of an Algorithmic State Machine chart.

Buffer unit. We chose a double-buffering scheme that allows one buffer to collect data and the other to empty its data into the SBC to maximize concurrency. The estimation of buffer size constitutes a critical design parameter. The buffer should be large enough to collect data over a given period of time without overflow and small enough to be emptied within the same time span.

Applying a queueing analysis normally carries out the estimation. Assume that the data arrives (at the buffer) at some rate R for servicing (it is transferred to the SBC) and that the service rate is the data-output rate T. Since the periodically serviced buffer can overflow if it is too small, our objective was to determine a buffer size that would eliminate such an overflow or loss of information. In the following, we determine the input and output rates and the distribution of the data.

The rate at which the system writes to the buffer depends upon the rate at which the Equal signal changes from state to state. The higher the frequency of change, the higher the input rate to the buffer. The worst case condition corresponds to a state change during every clock cycle. With a clock frequency of 10 megahertz, the input rate is 1 occurrence per 100 nanoseconds. In addition, we assume that this input rate follows a Poisson distribution.[4]

Output rate. The nature of the underlying communication path largely determines the output rate for emptying the buffer. We assume the path to be simplex, that is, the data transfers in only one direction along this dedicated path. The parallel port of the SBC serves as the interface. The time required to transfer the data from this interface to the local memory of the SBC can be determined from the SBC's constituents.[5] This process involves an 8225A programmable peripheral interface and an 8259A priority-interrupt controller. We calculated the time required to read a byte of data as 3.225 microseconds.[6] We therefore estimate 4µs as the nominal service time required to transfer a unit of data.

Buffer size estimate. Let N be the buffer size, R the mean input rate, and T the transfer time per byte. The SBC takes NT time to flush the buffer. Given two buffers and an input rate that follows Poisson statistics, we define $P(N)$ as the probability that the second buffer has N arrivals in time T. That is,

$$P(N) = [(RT)^N \exp^{-RT}]/N!$$

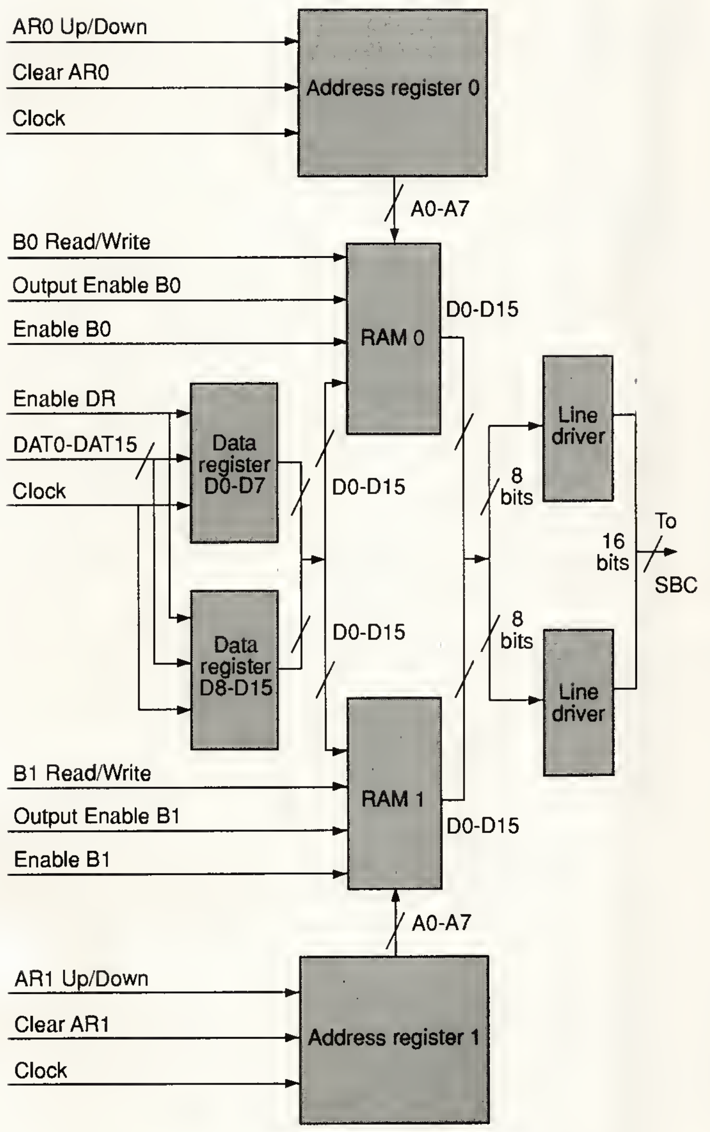
To eliminate overflow, we need to ensure that the probability of N or more arrivals in the given time is at the minimum, say, less than 0.15 percent. That is,

$$P(N) + P(N+1) + P(N+2) + \ldots < 0.0015.$$

Determining buffer size N is based on the assumption that the probability of the second buffer overflowing is less than the chosen value, given that it takes NT time to flush the first buffer. Using $T = 4$ µs and $R = 1/(100$ ns), a good value of N turns out to be 60.

Figure 7 provides a few details of the actual implementation. The buffer has four distinct components: the data registers, RAMs, address registers (ARs), and the line drivers. An AR starts at zero and counts positively every time a word has to be written. When the count reaches its maximum value, the buffer fills, which initiates a corresponding signal to the control unit. An AR is in the *up count* mode during this operation.

Once the data-gathering operation completes—or



A0-A7	Address register lines	B1	Buffer 1
AR0	Address register 0	D0-D15	Data lines
AR1	Address register 1	DAT0-DAT15	Data register lines
B0	Buffer 0	DR	Data register

Figure 7. Buffer unit.

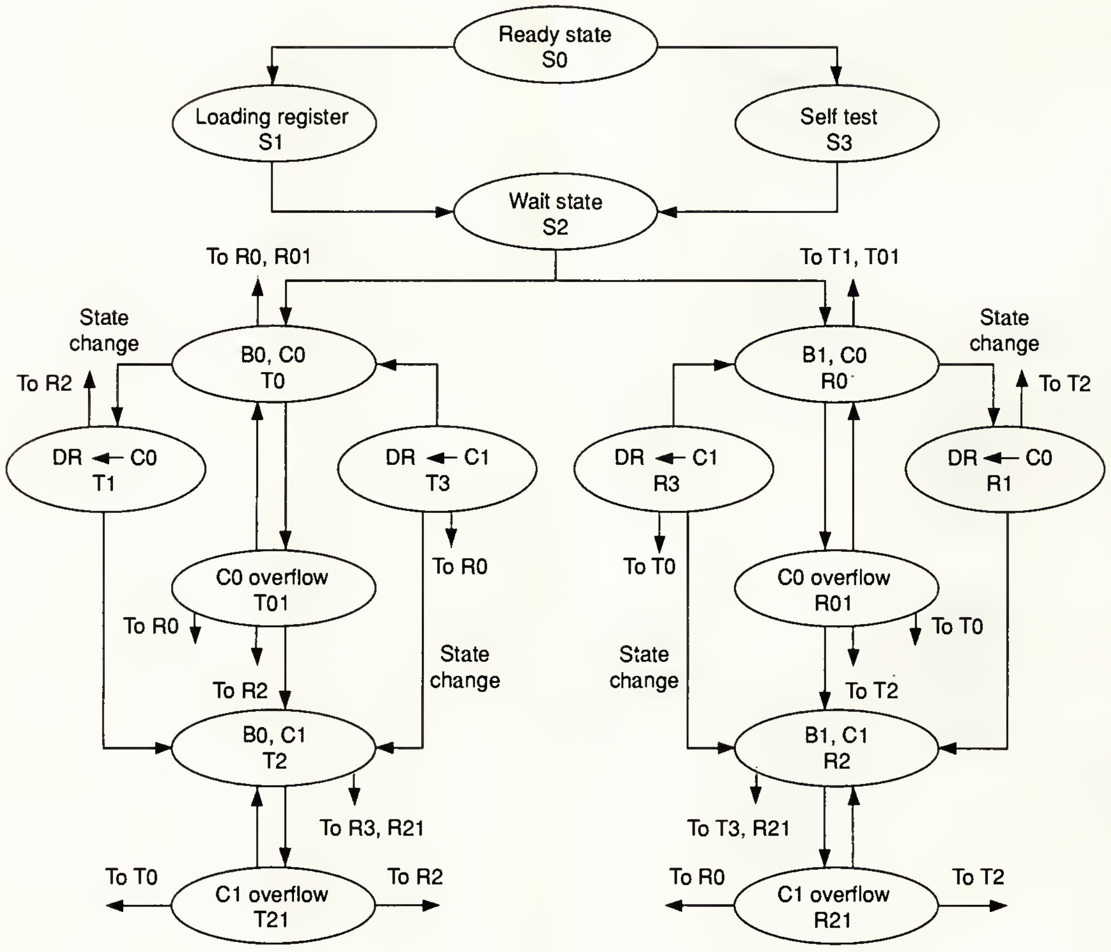
the buffer fills—the system maintains the value of the AR until a data-send operation begins. When the data is ready to be sent to the SBC, the control unit activates the read line of the corresponding RAM buffer and enables the line driver. The AR switches to the *down* mode. The unit routes the data read from the RAM to the parallel port of the SBC via the line driver. It also sends a strobe signal to the parallel port of the SBC to latch the data onto the port. The SBC acknowledges with the Input Buffer Full (IBF) handshake signal. When the monitor recognizes (from IBF) that the latched data has been cleared, it decrements the corresponding AR and reads the next word. This process continues until the AR reaches a value of zero, which indicates that the buffer is empty. The control unit then sets the appropriate status bits.

Control unit. This unit (shown earlier in Figure 3) coordinates the actions of the other units. It receives instructions from an external source, interprets them, and initiates actions on the monitor board. The control unit responds to the following instructions:

- *Reset.* The control unit initializes all units to a known state.
- *Loading registers.* The system loads programmed data into the pattern register, the don't-care mask register, and the timer (which times the duration of monitoring).
- *Start.* Once the SBC has loaded the registers, it issues a Start command to the monitor. The control unit then enables the event filter, edge detector, counter, and buffer unit as required. Once the specified time duration elapses or the system identifies an error such as buffer overflow, the control unit stops all units and raises an interrupt line to inform the SBC.
- *Indefinite time.* If the SBC requires that the monitoring continue for an indefinite period of time, it does not write into the timer. Instead, it issues the Indefinite Time command to the monitor in place of the Start command. The control unit then activates all units until an error arises or it receives a Halt command from the SBC.
- *Halt.* This instruction overrides any previous command, stops system monitoring, and also brings the monitor to an input-accepting state.
- *Read status.* An SBC uses this instruction to determine the status of the monitor by reading the contents of the status register.
- *Send data.* Once the data is collected in the buffer—or the buffer has become full—it needs to be emptied. The SBC issues a Send Data command to the monitor to begin this process. Data transfer can occur in parallel with the rest of the monitor operations.
- *Self test.* The monitor can test itself (see the section on this unit).

The state diagram in Figure 8 on the next page depicts the overall control-unit operations.

The control unit uses erasable programmable read-only memory. Four EPROMs implement a horizontal microcode to generate all control signals. The EPROMs function differentially to



B0	Buffer 0	DR	Data register
B1	Buffer 1	R,S,T	
C0	Counter 0	(with no.)	State name
C1	Counter 1		

Figure 8. State diagram of the control unit.

- service data-send operations,
- process inputs from the SBC,
- collect data into Buffer 0, and
- collect data into Buffer 1.

Interference. Removing the effects of interference from the monitor or the data is an important control function. An active monitor generates interference. At that time, the SBC sends instructions to the monitor via the Multibus. Because this signal is not part of the host system and is introduced only when the monitor is present, we do not take it into account. Disabling all units of the monitor when the SBC tries to access it removes the interference.

Self-test unit. This unit essentially simulates the activity on the bus. It loads a pattern into the registers and starts the monitor. It then generates patterns and applies them to the bus register. The event filter compares and collects data as in a normal operation.

At the end of the specified time, the self-test unit compares the data collected by the monitor in the buffers with the prestored data. If the two match byte-for-byte, the monitor functions properly. If the match is defective, the monitor could enter an internal, infinite loop. A watchdog timer[7] that raises an interrupt after a certain period of time can force the monitor out of the loop.

Applications

Monitors are essential tools for performance evaluation and management in the field of distributed processing. Because monitors are functionally multidimensional, we see applications in several of the following areas.

Event recorder. The recording of events—the primary role of a monitor—allows event simulation at any arbitrary point of time. Specifically, the time mode of operation provides straightforward reconstruction of events. Event recording and playback are also very conducive to catastrophe studies (detection as well as prevention).

Debugging tool. Debugging a multiprocessor system is a complex process. A monitor can assist greatly by isolating problem areas as it seeks specific faults or patterns.

Fault-tolerance aid. One can design a monitor to detect failures in nodes as well as in communication links. Our system functions as both a cluster monitor and a network monitor in Ted. A control processor subsequently takes corrective actions based on the monitoring result. A playback of events also allows recovery or restarting from the point of failure.

Universal monitor. We designed and tested the present monitor for the Multibus architecture. However, it is suitable for any arbitrary bus, with a few modifications. The captured data has to be interpreted for each type of bus architecture. Once this is achieved, one can connect clusters of different architectures. The universal monitor can evaluate, interpret, and manage the cluster. Thus one can evaluate different bus features with the same instrument.

We have designed and implemented a hardware monitor to handle the monitoring activities within a cluster in the Ted system. By using specified patterns and don't-care masks, the system can detect accesses to selected data, addresses, or blocks of addresses. This function helps monitor events such as the access or usage of a memory location or a group of mailbox addresses. It also determines the amount of time consumed by the performance of specific operations.

We plan to accomplish active monitoring in which an external computer system connected to an Ethernet network controls a Ted system.

Acknowledgments

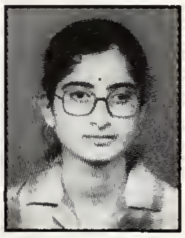
We thank Bohdan Bodnar and George D. Kraft for the suggestions and support provided at every stage of this project.

References

1. D. Ferrari, G. Serrazi, and A. Zeigner, *Measurement and Tuning of Computer Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1983.
2. D. Jacobson et al., "A Master/Slave Monitor Measurement Technique for an Operating Ethernet Network," *IEEE Network*, Vol. 1, No. 3, July 1987, pp. 40-48.
3. "Intel Multibus Interfacing," Appl. Note AP-28a, Intel Corp., Santa Clara, Calif., Jan. 1979.
4. E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems*, Academic Press, London, 1980.
5. *iSBC 86/14 and 86/30 Single Board Computer Hardware Reference Manual*, Intel Corp., Santa Clara, Calif., 1982.
6. A.C. Liu and R. Parthasarathi, "Hardware Monitoring of a Multiprocessor System," tech. report, Distributed Processing Laboratory, Illinois Institute of Technology, Chicago, 1988.
7. G.D. Kraft and W. Toy, *Mini-micro Computer Hardware Design*, Prentice-Hall, Inc., 1976.
8. D. Thomson, "Helping Data Managers Find Their Voice," *Data Communications*, Vol. 14, No. 5, May 1985, pp. 111-123.

An-Chi Liu is a professor at the Feng Chia University, Taichung, Taiwan, where he chairs the Department of Information Engineering. He was a faculty member at the Illinois Institute of Technology when he authored this article. Liu's research interests include distributed processing and computer networks.

Liu received the BSEE and MS degrees from the National Chiao Tung University in Taiwan and the PhD degree from the University of Illinois at Chicago.



Ranjani Parthasarathi is a development engineer in the Business Communications Group of Indchem Electronics in India, where she participates in application-specific IC (ASIC) design for processing Indian languages.

Parthasarathi received the BS degree from Madras University in India and the MS degree from the Illinois Institute of Technology, where she coauthored this article. She is a member of the IEEE.

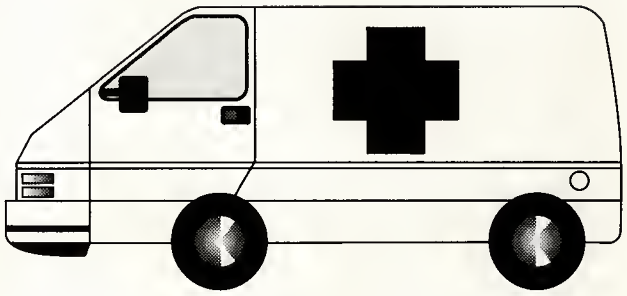
Questions concerning this article can be directed to An-Chi Liu, Department of Information Engineering, Feng Chia University, 100 Wenhua Road, Taichung 40724, Taiwan, Republic of China.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153 **Medium** 154 **High** 155

The P1073 Medical Information Bus



A group of three proposed IEEE standards specifies a network for collection and measurement of bedside data.

David F. Franklin
Southern College of Technology

David V. Ostler
Motorola/Emtek Health Care Systems

The P1073 Medical Information Bus (MIB) family of three proposed IEEE standards provide vendor-independent interconnection and interoperability of medical devices and computer systems. The MIB uses a layered network architecture that is compatible with International Standards Organization/Open Systems Interconnection (ISO/OSI) specifications.[1] The proposed standards primarily focus on the acute care environment at the patient's bedside, particularly when this environment is in the intensive care area of the hospital. However, the MIB is suitable for use in other areas such as the operating room or the general ward. The bus can also interface to any general clinical equipment for simple data communication and control purposes.

The proposed standards provide for a specialized local area network between one or more groupings of medical instruments and a host computer system. The host system can operate from the bedside or from a remotely located area, where it can be accessed via a conventional LAN.

The MIB differs from a conventional LAN due to the unique requirements of the medical environment. Frequent changes in the number and types of devices at the bedside require automatic reconfiguration of the network as devices are connected or disconnected. Medical users that participated in our project required that the devices and systems interface with an absolute minimum of user interaction. The MIB network automatically detects device connections, establishes communications, and identifies devices. The user merely plugs in a data cable and switches on the device.

P1073 also specifies the Medical Device Data Language. The MDDL communicates measurement data, status, and control information between the host system and the medical instruments.[2] Clinical data can involve much more than simple numeric values, particularly if the device is aware of parameters concerning the context of the measurement. These parameters can include measurement units, sites, and methods, as well as the time the measurement took place.

An earlier version of this article appeared in the *1989 Proceedings of the 8th Annual International Phoenix Conference on Computers and Communications.*

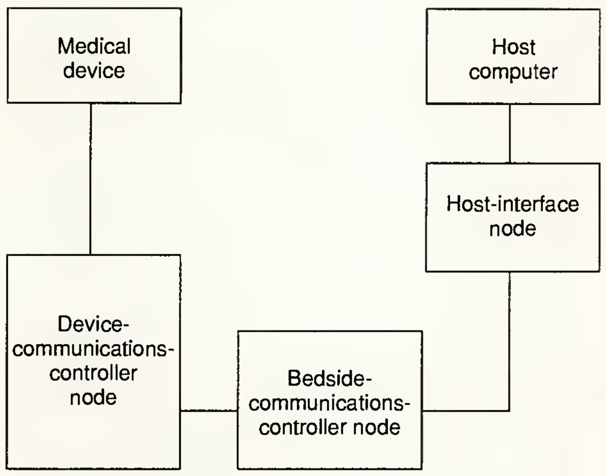

Medical device
Host computer
Host-interface node
Device-communications-controller node
Bedside-communications-controller node

Figure 1. The MIB network nodes.

Intelligent monitoring devices with numerous modes of operation and alarms communicate not only the clinical data collected by the device but also status data concerning the current state of the device itself. Therapeutic instruments such as ventilators and closed-loop infusion systems require that device-control information be included in the list of communications requirements.

The MDDL provides operations for exchanging clinical data in context, including both the measured values and those attributes that describe the measurements known to the device. The MDDL also provides mechanisms to handle the commnications necessary for monitoring device status and handling device-control information. Altogether, the MDDL provides a comprehensive, relatively complex set of device-related measurement operations necessary for host-system and intelligent-device functioning. However, one can also streamline the MDDL into a very compact form for efficient implementation on simple measurement devices that supply only numeric data without any context.[2]

Nodes on the network

The MIB requires three different types of logic nodes to form an operational network (Figure 1). A medical device interfaces to the MIB through a device-communications controller. This DCC connects to a bedside-communications controller to form a local subnetwork at that location. The BCC functions as a concentrator or multiplexer between the bedside medical device and the host-interface node.

The DCC contains hardware for a plug-compatible interface and software for reliable communications between the medical device and the MIB. The BCC contains hardware—and the minimum control software—for reliable data transfer between the DCC and the host interface. The BCC automatically detects device connections and disconnections at the bedside. The host-interface node is a software interface between the MIB and the host computer that is responsible for collecting data from the bedside.

The MIB's flexibility allows either workstations or monitors at the bedside to serve as hosts. One or several MIB nodes can reside in the same physical instrument or system. Intelligent monitoring instruments can perform BCC functions, or workstations at the bedside can perform both host and BCC functions. Or, with an enhanced BCC, a conventional LAN connection can access remotely located minicomputer hosts. Just as many DCCs can connect to one BCC, multiple BCCs can communicate with a single host system that serves multiple bedsides (see Figure 2 on the next page).

One can build DCCs to handle multiple devices such as multichamber infusion pumps, or devices with varying parameters such as monitoring systems with plug-in modules. Generic add-on DCCs can interface existing simple measurement devices to the P1073 network.

P1073 describes a method for measuring the data load at a bedside. This procedure aids proper configuration of an MIB network for a specific installation. We measure the amount of data that a particular device places on the network in MIB loading units. We describe devices and their DCCs by the number of loading units they need for operation on the network. Similarly, we rate BCCs and hosts by their processing capacity in terms of loading units. We can size and produce BCCs in different capacities according to the load requirements of different bedside environments (operating rooms, intensive care units, or the general ward). The hospital's biomedical or clinical engineers determine the number of loading units needed at a particular bedside, along with the number of MIB outlets.

Family of proposed standards

Here we present the general specifications of each document in the P1073 group. A more technical description follows in the next section.

P1073.1, Medical Information Bus: Network Architecture and Application Interface. This document describes the upper three OSI layers—application, presentation, and session—as well as the overall network architecture. The application layer specifies the use of ISO protocols for remote-operations and association-control service elements (ROSEs[3,4] and

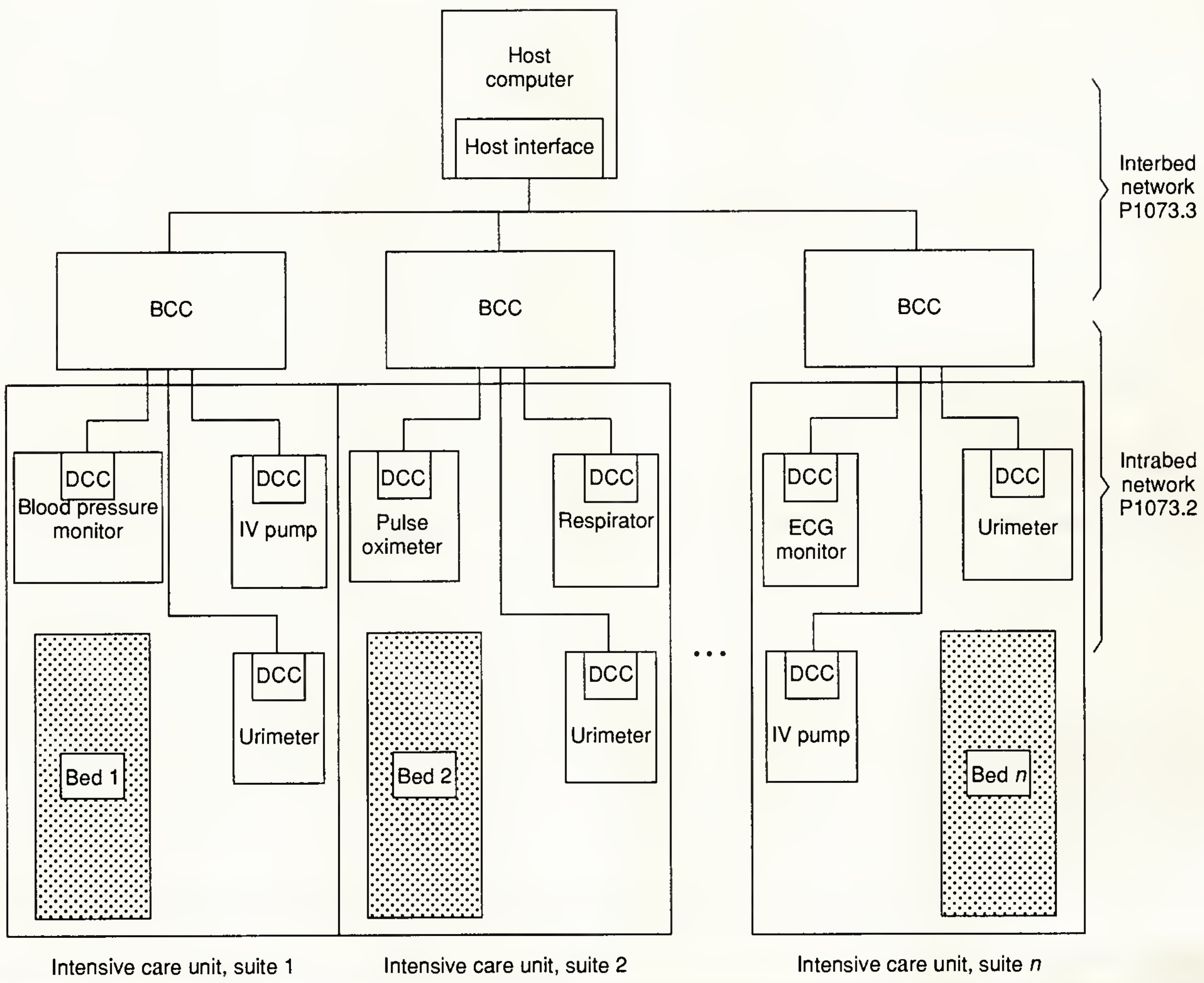


Figure 2. The MIB interbed and intrabed networks. (Courtesy of Tim Kelly, C.R. Bard, Inc.)

ACSEs[5,6]), as well as medical device application service elements (MDASEs). (See Figure 3 and Table 1.)

P1073.2, Medical Information Bus: Bedside Communications Subnetwork. This proposal describes the subnetwork that provides the transport service between the DCC and the BCC. This subnetwork consists of the lower four OSI layers: transport, network, data link, and physical. The inactive transport and network layers do not allow packet assembly or multiplexing. The data-link layer uses a specified subset of high-level, data-link control (HDLC[7]) procedures and a strict polling protocol to provide reliable link service. (See Figure 3 and Table 1.)

P1073.3, Medical Information Bus: Gateway Function Between a P1073.2-compliant Network and an ISO 8072/8073-compliant Network. This document describes the access function between a BCC and a host that is located on a conventional LAN (see Figure 4 and Table 2 on page 56). The specifications of this proposed standard are required only in the case of a remote host. A local host network need only use P1073.1 and P1073.2 specifications as shown in Figure 3. We assume that connection-oriented transport services are available for a host on a conventional LAN. ISO 8072[8] and ISO 8073[9] describe the services and protocol for reliable end-to-end communications at the transport layer.

P1073.3 provides for a gateway from a P1073.2 subnetwork to a LAN. Current efforts (still in an early stage of development) focus on the issues that involve internetworking MIB, as we discuss later. These issues also include supporting multiple, simultaneous conversations between devices and hosts (multiple-association or application-level gateways).

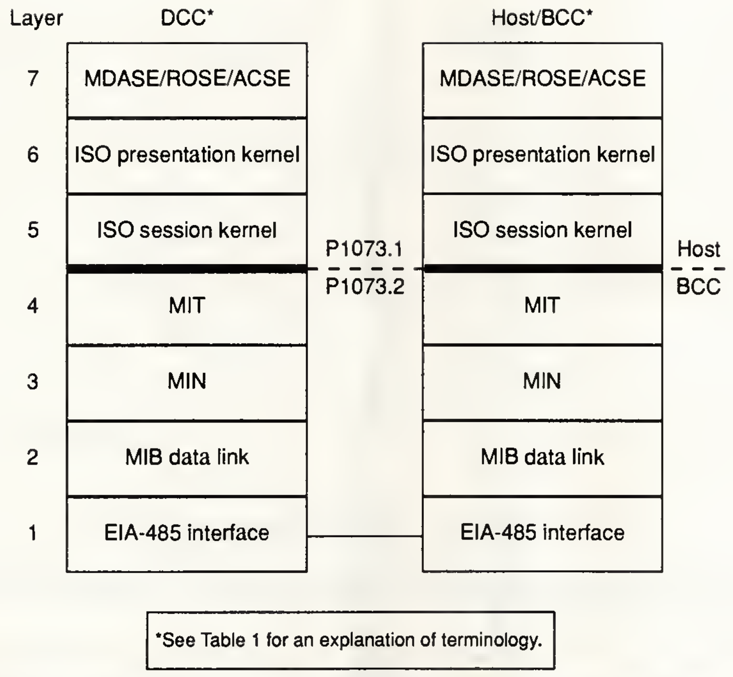


Figure 3. The MIB protocol stacks.

Table 1.
Summary of MIB protocol stacks.

Layer	Name	Description
P1073.1 (upper layers)		
7	ACSE/ROSE	The kernel subset of ISO 8649/8650 (ACSE) and ISO 9072 (ROSE) application layer services and MDASE
6	MIB presentation	The kernel subset of ISO 8824/8825 presentation-layer services
5	MIB session	ISO 8326/8327 session-layer kernel subset
P1073.2 (lower layers)		
4	MIB inactive transport (MIT)	MIT service primitives and parameters identical to those for ISO. Most parameters defaulted in the MIT protocol.
3	MIB inactive network (MIN)	A functionally inactive layer, with a control header specified for use in future revisions.
2	MIB data link	A specified subset of HDLC, using two-way alternate, normal-response mode, with the BCC as the primary station and the DCCs as the secondary stations.
1	EIA-485	Data transfer using Electronics Industry Association EIA-485 interface[10] at a signaling rate of 375 Kbps with NRZI encoding. BCC provides 12VDC ± 2V current limited to 250 mA of power to DCC. Specifies a unique 6-pin connector and a shielded cable with three twisted-wire pairs.

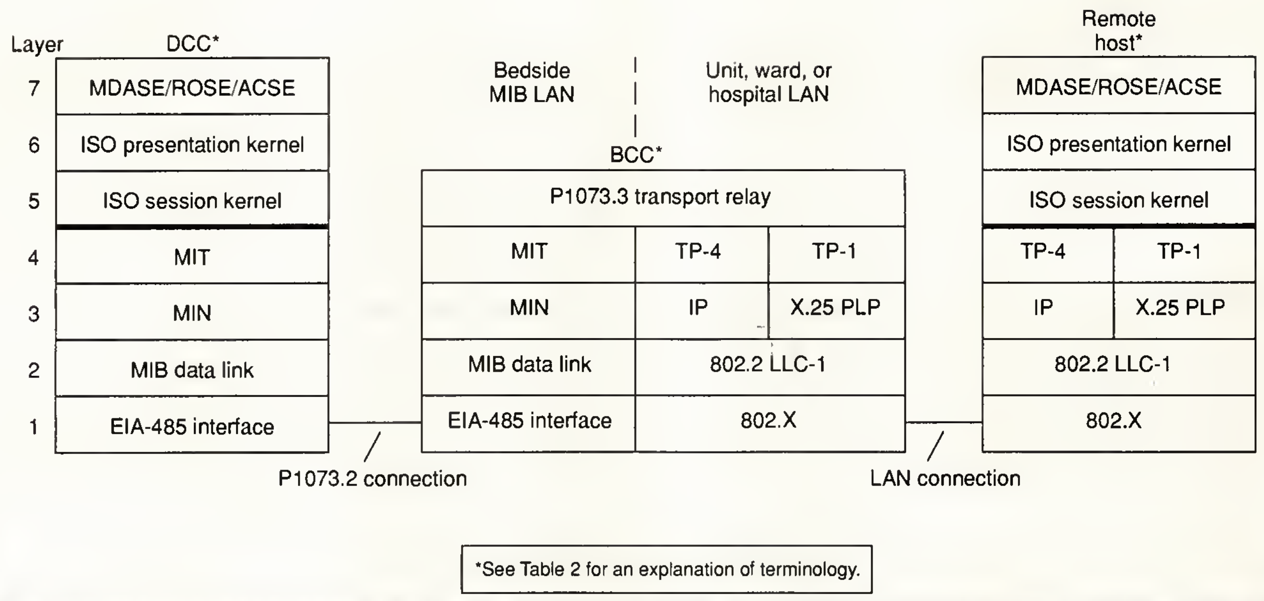


Figure 4. The MIB network with a remote host.

Table 2.
Summary of LAN protocol stacks.

Layer	Name	Description
4	TP-4	Transport protocol class that provides reliable data transfer over an assumably unreliable network (one that does not guarantee a delivery) such as IP (see below).
4	TP-1	Transport protocol class that provides minimal data-transfer services over an assumably reliable network, such as X.25 (see below).
3	IP	Internet protocol that provides basic network-layer, data-transfer services (including services between different networks) without guaranteed delivery.
3	X.25 PLP	X.25 packet-level protocol that provides guaranteed network services at the layer data-transfer level. Subject to resets that must be handled by the transport layer.
2	802.2 LLC	ANSI/IEEE Standard 802.2 (ISO/DIS 8802-2)[11] on logical link control that provides both connection-oriented and connectionless data-link procedures over a variety of different transmission media.
1	802.X	Any of the family of ANSI/IEEE/ISO 802[12-14] transmission media that specify media-access control protocols and physical layer encoding and transmission characteristics. Includes Carrier Sense Multiple Access with Collision Detection (CSMA/CD), token-ring, token-bus, and fiber-optic interfaces for distributed data.

Technical information

Here we discuss the salient technical features of each proposed standard at its current stage of development.

P1073.1. As mentioned, this proposal discusses the overall MIB network and layers 5, 6, and 7. It differentiates between required and optional features, identifies unique characteristics and requirements of the network, and discusses functional requirements of the user interface.

We are currently defining MDDL for possible use in a MDASE in the application layer (see Figure 5). A MDASE can make use of the ROSEs and ACSEs discussed in the ISO Draft International Standards 9072-1 and 9072-2[3,4] (for model, notation, and service defini-

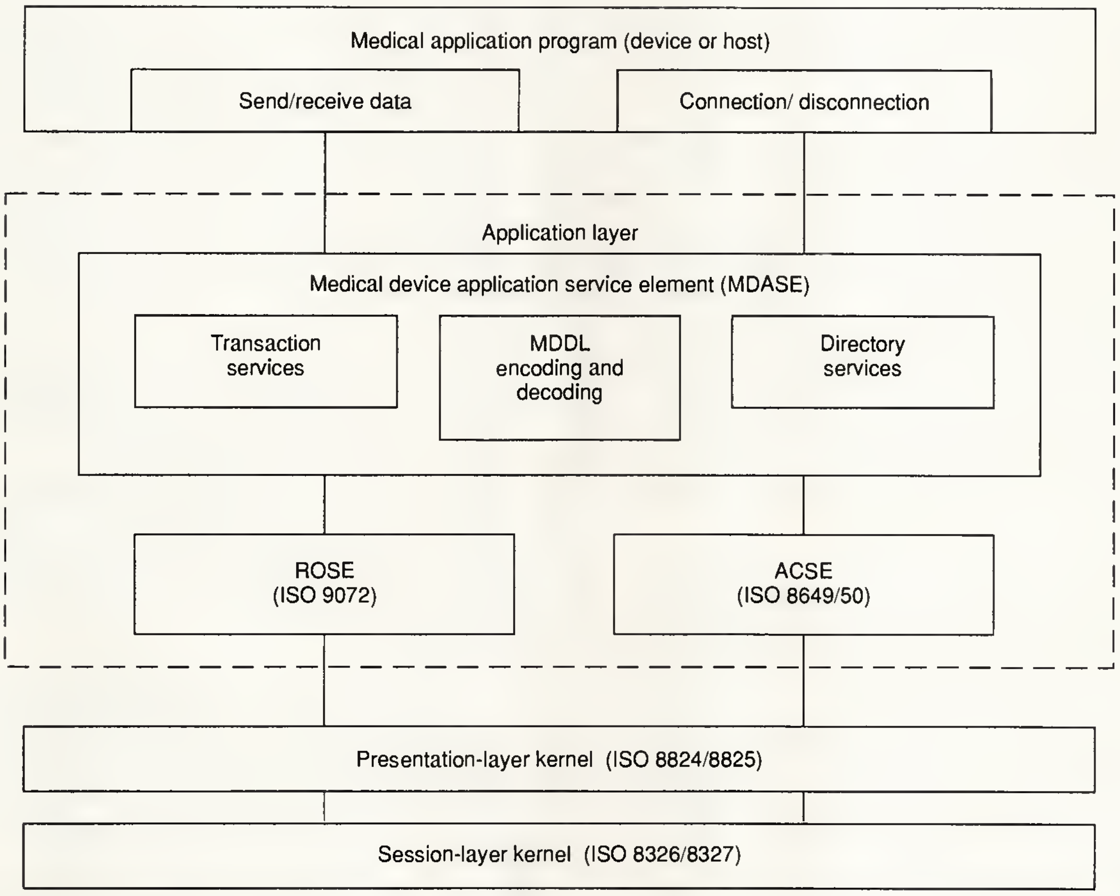

Medical application program (device or host)
Send/receive data
Connection/ disconnection
Application layer
Medical device application service element (MDASE)
Transaction services
MDDL encoding and decoding
Directory services
ROSE (ISO 9072)
ACSE (ISO 8649/50)
Presentation-layer kernel (ISO 8824/8825)
Session-layer kernel (ISO 8326/8327)

Figure 5. The MIB upper layer network architecture.

tions for remote operations) and ISO Standards 8649/50[5,6] (for the service definitions and protocol specifications of ACSEs). MDDL previously had solely consisted of a structured set of ROSE operations, with parameters, attributes, and data types written in the Abstract Syntax Notation (ASN.1) specification.[15,16]

MDDL has recently been redesigned as a flexible, extensible messaging system. It first specifies the rules for forming (or parsing) the messages. These rules include specifications of required and optional fields. Hence, MDDL specifies a structure or template for the messages.

Second, MDDL rarely restricts the actual contents or meaning for a given field in the message. However, MDDL medical-device messaging specifications define the required fields and default values for each recognized class of medical device (monitors, ventilators, infusion pumps, etc.). Default values specify established terminology and encoding schemes, where these exist. For example, Systematized Nomenclature of Medicine, or Snomed, codes[17] can serve as the default specifications for the anatomical location of a measurement. International Medical Device Codes can provide a system for identifying the devices.[18] Systeme International units can serve as the defaults for units of measurement.

P1073.1 specifies the kernel subsets of several ISO standards for the presentation and session layers. (ISO 8824/25[15,16] concern ASN.1 and its basic encoding rules, while ISO 8326/27[19,20] specify basic connections-oriented service definitions and protocols for the session layer.) These layers are required for compliance on the DCC and the host-interface nodes of the MIB network.

Other specific user-interface requirements include positive visual indication of a connection to the MIB network and an operator button on the device to signal intentional disconnections. A two-digit display on certain DCCs allows operators to discriminate between multiple devices. Operators can then distinguish between multiple infusion pumps for fluid-identification purposes, for example.

P1073.2. This proposal specifies the lower level architecture for the MIB network. In addition to the services and protocols for layers 1 through 4, the document specifies the hardware, cabling, and connectors necessary for plug compatibility between DCCs and BCCs.

We optimized the P1073.2 subnetwork for use in the medical environment. The proposal focuses on providing fast, reliable communications while also detecting device connections/disconnections. The bedside subnetwork consists of a star point-to-point physical topology (Figure 2). As a result, network addresses are location-dependent within a P1073.2 subnetwork. The BCC associates every connected device with a particular bedside (patient). During connection, the device specifies quality-of-service (QOS) parameters for only the maximum and minimum throughputs required in each direction. These parameters translate into MIB loading units during the connection process, as described later. The subnetwork defaults all other QOS parameters.

Transport layer. P1073.2 provides transport-layer services between a DCC and a BCC. The MIB inactive-transport (MIT) layer service primitives appear similar to ISO 8072/8073 transport services, but do not strictly comply with ISO 8072/8073. MIT does not support packet assembly and disassembly and contains a simplified set of active parameters in its connection-service primitives due to null address fields and other defaulted QOS parameters. A single, unconfirmed, data-transmission-request primitive and an indication primitive exist. Disconnection service, which is unacknowledged by the bus, destroys any queued packets. Only a DCC can initiate connection requests. Both DCCs and BCCs can initiate data-transfer and disconnection services.

Network layer. Although the MIB inactive-network (MIN) layer is functionally inactive, it does have service primitives for connection, data transfer, and disconnection. The layer accepts service data units from the transport layer and adds a 1-byte, protocol-control information (PCI) field (as does the MIB transport layer). The PCI's presence in the network and transport layers allows backward compatibility in the case of a new MIB lower layer protocol set defined in the future with mulitplexing or packet-assembly/disassembly functions.

Data-link layer. This layer provides service primitives for connection-oriented, point-to-point data transfer. The protocol uses an optimal subset of the two-way alternate (TWA) normal-response mode (NRM) of HDLC elements of procedure ISO 4335.[7,8] (This standard describes high-level data-link control procedures and consolidation of their elements in information processing systems.) The BCC functions as the primary station, and all DCCs function as secondary stations. The BCC employs a polling protocol to detect device connections/disconnections.

Connection requests are negotiated through the exchange of control packets.[21] Maximum-information field length is negotiated through the use of MIB loading units. The BCC polls empty ports by sending control-packet frames that offer the maximum loading units available at that port. The connection request at the DCC specifies the loading units needed by the device. If the units offered are adequate, the DCC sends a control-packet frame in response to the control-packet poll. During control-packet frame exchange, the system sets the transmission window size to 1 and uses a sequence-number modulus of 8 for each end of the connection.

Data-transfer services use information and supervisory frames. Either end can issue a disconnection request, which is destructive to any queued packets and is unacknowledged. The poll/final bit functions as the token in the half-duplex transmissions.

The P1073.2 proposed standard specifies strict response times and transmission-retry limits. Failure to respond to a BCC poll—or the absence of a poll at the DCC within a 1-second period—breaks the connection. The detecting station must return the port to the disconnected status.

Physical layer. This layer uses EIA-485 specifications with nonreturn to zero inverted (NRZI) encoding at 375 Kbps. The proposal specifies a unique 6-pin connector and a shielded 6-wire cable. One wire pair provides half-duplex data transmission. Another pair supplies 12VDC ± 2V current limited to 250 milliamperes for optional use by the DCC communications hardware. An optional encoding mechanism on the third wire pair provides for time alignment between instruments.

P1073.3. For local hosts—like an intelligent workstation at the bedside—the P1073.1 host interface uses the transport services provided by the P1073.2 subnetwork. A remote host that is accessible through a conventional LAN uses the transport services of that LAN. Before remote hosts can communicate with a P1073 network, at least two major issues need resolution. One involves the internetworking between P1073 and another network. The second involves whether to support only single associations (one-to-one) or multiple associations (one-to-many) between a single device and its host(s).

The original P1073.3 project authorization addresses only the first issue, and then only in a rigidly specified manner. The following paragraphs discuss this version of P1073.3. However, note that several other architectural solutions are possible (bridges and routers, for example). The multiple-access problem, having been recognized as an issue, awaits completion of P1073.1, which addresses the single-association case.

P1073.3, Version 1. Communicating with a remote host requires that an enhanced BCC provide a relay function between the transport services of the P1073.2

bedside subnetwork and those of the remote host. In addition to the usual P1073.2 functions and hardware, this enhanced BCC contains hardware and software associated with the LAN located between it and the remote host. The BCC must also contain control software for mapping the services and parameters between the two networks. As currently authorized, P1073.3 specifies the relay function between the transport services of a P1073.2 network and an ISO 8072/73-compliant network.[8,9]

In the latter network, the calling transport service-access point (TSAP) is the DCC TSAP, and the called TSAP is the host TSAP. The host TSAP is stored in nonvolatile memory within the BCC during initialization. The transport selector address field of the DCC TSAP is the physical port number of the device connection inside the BCC.

When a P1073.3 BCC receives a connection indication from the P1073.2 transport layer, the BCC must initiate a transport-connection request on the ISO 8073 network. When it receives a connection confirmation from that network, the BCC forwards a connection response to the DCC on the P1073.2 subnetwork. The QOS parameters passed between the two networks specify the level of needed service. P1073.2 QOS parameters contain almost all of the default values needed to tightly specify the required service. The BCC must map P1073.2 default values onto ISO 8073 parameters, and vice versa, in a manner that is consistent with its knowledge of the requirements of the two networks.

After the connection is established between the two networks, the BCC relays data indications received from either network as data requests to the other network. Disconnection indications from either network become a disconnection request to the other network.

The current first rough draft of P1073.3 contains several options for BCC initialization, including operator entry, host-initiated connection to the BCC, and BCC broadcast messages to find a host.

Status of P1073

Although early prototypes have existed for several years,[22] the design of the MIB network has evolved and changed considerably due to input from various sectors of the healthcare professions and industries.

As of this writing (August 1989), the working group is performing the final editing of the P1073.2 draft document. Different parties have independently built at least two P1073.2 subnetwork prototypes, which were demonstrated as stand-alone systems. The last remaining step is demonstration of the interoperability of the two prototypes. We expect the final draft document to be passed to the IEEE for formal balloting and public comments before the end of this year.

The group is currently revising the draft of the P1073.1 document to incorporate reference models for the network-architecture, virtual-device, MDDL-language, and MDDL-messaging specifications—along with the services and protocols for the upper three layers. We expect a substantive draft of P1073.1 by the first quarter of 1990. We have already demonstrated one significant early prototype that contains MDDL and the P1073.1 upper layers. We plan to demonstrate additional prototyping of the upper layers later this year.

P1073.3 has proceeded through the rough-draft stage. The internetworking and gateway-access issues previously mentioned are under study. However, committee efforts currently focus on the completion of P1073.1 and P1073.2. (Although P1073.3 is optional, P1073.1 and .2 are mandatory for compliance.)

The professional community has expressed considerable interest in the MIB, both locally and internationally. We plan to hold a P1073 working group meeting at a major international healthcare organization conference in 1990. We have established liaisons with other national and international healthcare standards groups.

The refinement of MDDL rests on additional public input. Initial groups of classes of devices and corresponding message sets are being defined for the first version of MIB. More focus groups would provide further definition and refinement of additional device classes and message sets. Like the current P1073 working group, these focus groups would consist of individuals from the healthcare and computing industries, clinical users, the biomedical and clinical engineering communities, and other interested parties.

Through a standard data interface and communications network, the Medical Information Bus could finally open the door to large-scale, automated collection and integration of clinical data into the electronic medical record.

References

1. *ISO 7498, Information Processing Systems—Open System Interconnection—Basic Reference Model*, International Organization for Standardization, Geneva, Switzerland, or American National Standards Institute, New York, 1984.
2. Jan Wittenber, "A Report on the Development of a Medical Device Data Language (MDDL) for the P1073 Medical Information Bus (MIB)," *Proc. Second Ann. IEEE Symp. Computer-Based Medical Systems*, IEEE CS Press, Los Alamitos, Calif., pp. 140-151.
3. *ISO 9072-1, Information Processing Systems—Text Communication—Remote Operations—Part 1: Model, Notation, and Service Definition*, International Organization for Standardization or American National Standards Institute, 1989.
4. *ISO 9072-2, Information Processing Systems—Text Communication—Remote Operations—Part 2: Protocol Specification*, International Organization for Standardi-

zation or American National Standards Institute, 1989.

5. *ISO 8649, Information Processing Systems—Open Systems Interconnection—Service Definition for the Association Control Service Element, 1988,* International Organization for Standardization or American National Standards Institute, 1988.
6. *ISO 8650, Information Processing Systems—Open Systems Interconnection—Protocol Specification for the Association Control Service Element, 1988,* International Organization for Standardization or American National Standards Institute, 1988.
7. *ISO 4335, Information Processing Systems—Data Communication—High-Level Data Link Control Procedures—Consolidation of Elements of Procedures,* International Organization for Standardization or American National Standards Institute, 1987.
8. *ISO 8072, Information Processing Systems—Open Systems Interconnection—Connection Oriented Transport Service Definitions,* International Organization for Standardization or American National Standards Institute, 1986.
9. *ISO 8073, Information Processing Systems—Open Systems Interconnection—Connection Oriented Transport Protocol Specification,* International Organization for Standardization or American National Standards Institute, 1986.
10. *EIA-485, Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems,* Electronics Industries Association, Washington, D.C., 1982.
11. *ANSI/IEEE Standard 802.2-1985 (ISO/DIS 8802-2), LANs: Logical Link Control,* IEEE CS Press, 1985.
12. *ANSI/IEEE Standard 802.3-1985 (ISO/DIS 8802-3), Carrier Sense Multiple Access with Collision Detection (CSMA/CD),* IEEE CS Press, 1985.
13. *ANSI/IEEE Standard 802.4-1985 (ISO/DIS 8802-4), Token-Passing Bus Access Method and Physical Layer Specifications,* IEEE CS Press, 1985.
14. *ANSI/IEEE Standard 802.5-1985 (ISO/DP 8802-5), Token-Ring Access Method and Physical Layer Specifications,* IEEE CS Press, 1985.
15. *ISO 8824, Information Processing Systems—Open Systems Interconnection—Specification of Abstract Syntax Notation One,* International Organization for Standardization or American National Standards Institute, 1989.
16. *ISO 8825, Information Processing Systems—Open Systems Interconnection—Specification of Basic Encoding Rules for Abstract Syntax Notation One,* International Organization for Standardization or American National Standards Institute, 1989.
17. *Systematized Nomenclature of Medicine (SNOMED),* College of American Pathologists, Northfield, Ill., 1982.
18. *Universal Medical Device Nomenclature System,* ECRI, Plymouth Meeting, Penn., 1989.
19. *ISO 8326, Information Processing Systems—Basic Connection Oriented Session Service Definitions, 1987,* International Organization for Standardizatio or American National Standards Institute, 1987.
20. *ISO 8327, Information Processing Systems—Basic Connection Oriented Session Protocol Specifications, 1987,* International Organization for Standardization or American National Standards Institute, 1987.
21. *ISO 8885, Information Processing Systems—Data Communication—High-Level Data Link Control Procedures—General Purpose XID Frame Information Field Content and Format,* International Organization for Standardization or American National Standards Institute, 1987.
22. William L. Hawley, "Clinical Implementation of an Automated Medical Information Bus in an Intensive Care Unit," *Proc. Twelfth Ann. Symp. Computer Applications in Medical Care,* IEEE CS Press, 1988, pp. 621-624.

David F. Franklin is an associate professor in Applied Computer Science at Southern College of Technology in Marietta, Georgia. His primary interests include clinical data-management systems, computer networks, and embedded real-time systems design. He has been a member of the P1073 MIB working group since its inception and is the current chair. He is also a member of the P1157 Medical Data Interchange (Medix) working group.

Franklin received a BS in industrial design and an MS in information and computer science from the Georgia Institute of Technology. He is a member of the IEEE, the IEEE Engineering in Medicine and Biology Society (EMBS), and the ACM.

David V. Ostler is a senior engineer at Motorola/Emtek Health Care Systems in Tempe, Arizona. His primary field of interest is the exchange of clinical information between medical information systems and medical devices. He is a member of the P1073 and P1157 (Medix) working groups. He also chairs the IEEE EMBS standards subcommittee.

Ostler received a BS in electrical engineering, a BS in computer science, and an MS in medical biophysics and computing from the University of Utah. He is a member of the IEEE, the IEEE EMBS, the ACM, and the IEEE Computer Society.

Questions concerning this article and the MIB project itself may be directed to David F. Franklin, Applied Computer Science, Southern College of Technology, 1100 South Marietta Parkway, Marietta, Georgia 30060.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 159	**Medium** 160	**High** 161

The Micon System for Computer Design

This CAD system quickly configures microprocessor-based computers with the help of three AI modules.

William P. Birmingham
University of Michigan

Anurag P. Gupta
Daniel P. Siewiorek
Carnegie Mellon University

Semiconductor technology provides tremendous opportunities for high processing power and reliable, low-cost computers in the office or laboratory. Many of these machines, ranging from PCs to superminicomputers, use commercially available microprocessor family components. The high-end microprocessors available today commonly support virtual memory and main memory caches. A full array of high-performance dedicated processors (graphics controllers and numerical coprocessors) and communication components complement these devices. One can construct a very sophisticated machine almost entirely from off-the-shelf components. Semiconductor designers have just begun to tap a well of possibilities that promises even more performance and functions in future chips.

All of this advancement, however, extracts a price. Sophisticated components require sophisticated hardware designers. High-performance processors that commonly run in the 16- to 25-megahertz range force designers to become experts in high-speed logic design and computer architecture.

The days of simple designs that consisted of a processor and a serial I/O interface are over. Today's smallest computer system contains—at a minimum—disks, buses, and graphics-terminal interfaces. Workstations must use a minimum of three levels of memory hierarchy to achieve maximum performance. In addition, such traditional extras as networking and graphics are standard equipment. All of this integration places a strain on hardware designers who are trying to keep abreast of a rapidly evolving technology.

In the marketplace—where product life cycles are measured in months—the competition continues to become more intense. Three forces impact the hardware designer: a reduced design time, the ceaseless demand for increased performance at a lower price, and a constantly evolving technology. It appears unlikely that these forces will abate of themselves. Therefore, designers need some assistance in the form of CAD tools. We consequently designed the microprocessor "configurer" system, or Micon, to provide support for computer hardware designers.

The objective of Micon is to reduce the time required to construct a hardware system. The approach to achieving this objective rests on two points:

An earlier version of this article appeared in *Proceedings of the 26th ACM/IEEE Design Automation Conference*, © 1989 ACM 0-89791-310-8/89/0006/0135 $1.50.

• capturing and disseminating design expertise in a format that actively assists designers, and

• providing a tool environment that supports all aspects of computer design.

Numerous benefits accrue from making design expertise commonly available in a computer program. The system reduces the learning curve that faces designers when a new component is introduced. It also expands the number of new components that a designer can consider. Other benefits include fewer design errors and less time spent in trial-and-error learning. These factors allow the designer more time to create alternative designs, or just to create a working design more quickly.

The creation of computer hardware includes not only logic design and physical design but also reliability analysis. A design environment that supports multiple expert systems allows the designer to single-handedly address the entire spectrum of design activities. Such an environment handles data consistency issues more efficiently and results in fewer design errors.

Micon employs a set of complementary technologies. Artificial intelligence provides design synthesis and the acquisition of design expertise. Databases provide consistent views of data to all tools. Networking allows the system to efficiently share common resources, such as the database, across many users of the system. Here we describe the architecture of Micon, provide examples of its use, and describe a set of experiments to test its viability. We begin with comparisons of Micon to other related design systems.

Comparisons

Design synthesis is the process of producing a design that meets some set of specifications. In general, the specifications are abstract relative to the details needed to build the design. The task of a synthesis system is to provide the missing information. The mechanisms used to implement the synthesis system vary as widely as the types of synthesis tasks. Here we discuss related synthesis work in digital domains, but work is also progressing in other domains such as analog circuit design[1] and mechanical engineering.[2]

Digital system synthesis is an area of particular interest in the CAD field. Interest has shifted from graphics-oriented, schematic-capture systems to fully automatic systems that can produce sophisticated, fully operational designs.

Much of the recent work in digital system synthesis centers around the development of application-specific integrated circuits (ASICs). (We use the term application-specific very loosely here to connote systems that range from simple controllers to special-purpose microprocessors.)

In a broad sense, two approaches exist in ASIC design, depending upon the type and abstraction level of the input and the resulting design. The first approach proceeds from a high-level behavioral description and ends with an IC. The types of designs produced typically include microprocessors and minicomputer CPUs. The System Architect's Workbench (SAW) project,[3] which evolved from the Carnegie Mellon University Design Automation (CMU-DA) system project,[4] synthesizes digital systems from a high-level behavioral language called Instruction Set Processor Specification (ISPS).[5] SAW's objective is to produce a very large scale integration (VLSI) IC that implements the machine described by the ISPS input program.

The synthesis process consists of transforming the behavioral description into a technology-independent dataflow representation called a value trace graph. The graph is then subjected to a set of complex operations that result in a data-path and controller design represented as a set of register-transfer-level (RTL) components. The RTL representation binds to modules in an implementation technology, such as standard cell libraries.

The second broad class of ASIC synthesis systems takes a lower level description of a system as input for implementation. The descriptions vary, but typically spring from any of the following:

• an RTL hardware description language,
• a set of Boolean equations, or
• state tables.

These tools produce sophisticated designs that are smaller than the systems previously discussed. Examples of these designs, which are used as building blocks of larger digital machines, include programmable logic arrays, state machines, and blocks of combinational logic. Logic synthesizers, which are typical of this class of tools, produce ASICs by using a set of predefined cells (for instance, standard cells or gate arrays). IBM has put forth a long-standing, successful effort in this area with the development of a production quality system called the Logic Synthesis System.[6] A number of other systems employ a similar paradigm to produce combinational logic and/or state machines.[7]

Designers are applying artificial intelligence to a wide range of synthesis problems. The appeal of AI is that it uses nonmathematically based representations (an example set of representations for digital systems is found in Bobrow et al.[8]). AI can also use domain knowledge to reduce the search inherent in many synthesis problems. The success of the Xcon/R1 project[9] illustrated the potential of AI technology for synthesizing problems. The Design Automation Assistant system,[10] part of the CMU-DA system project, used domain knowledge to aid in the generation of an IBM System 370 design. The VLSI Expert Editor system[11] is an interactive knowledge-based editor that assists a user in traversing a hierarchy of functional blocks to

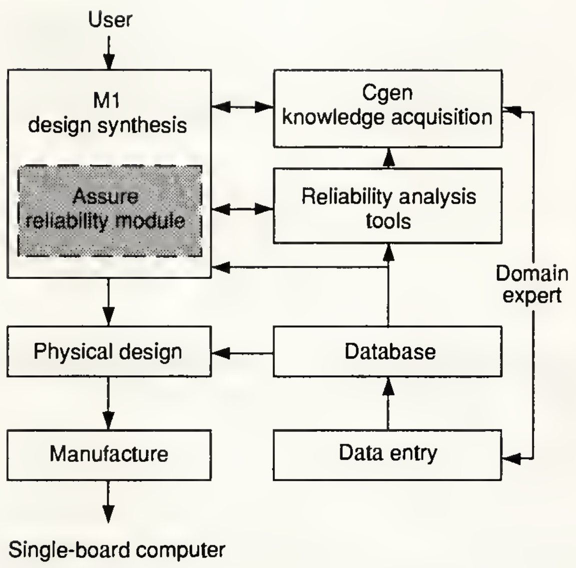


Figure 1. The Micon system.

create an N-channel, metal-oxide semiconductor implementation of a design.

A survey of the literature reveals that the design of pieces of computers (for instance, CPUs)—rather than complete computer systems—has driven most of the current synthesis research. However, as design tools become more proficient, system-integration issues will gain prominence. The Micon project concerns the synthesis of systems through the configuration of existing components. This approach emphasizes matching the interfaces between components (and in this way is similar to the Critter system,[12] although the intentions and implementations of the systems are very different).

The Micon project originated with M0 (a recently dubbed name),[13] a knowledge-based system that assessed the feasibility of automated single-board computer synthesis. This assessment involved commercially available microprocessors. M0 embodied, in a more primitive form, many of the concepts used in the design of our final M1 synthesizing tool. Several microprocessor families were hard-wired into the system. M0 produced a number of designs. One of them, based on the Z80, resulted in a working single-board computer. M0 also originated the idea of templates for representing structural information for synthesis. The system, however, had a number of drawbacks that limited its general applicability (see Gupta[14] for more details). These drawbacks led to the development of M0.5.[14] Even though M0.5 was not fully implemented, the ideas acquired from the project were essential to the final development of M1. In particular, M0.5 introduced component abstraction, which enables M1 to have a much larger degree of freedom in choosing components and thereby creates a larger number of interesting designs.

The Micon system

We originally developed Micon to design small computer systems. (We are now applying Micon to design problems in other domains such as design environments, or frameworks, and mechanical engineering.[15]) The computers designed by Micon consist of the following subsystems:

- *Processors.* The designs employ a microprocessor as the CPU. Presently, Micon can design with the Motorola MC6809, MC68008, and MC68010 and the Intel 80386.
- *Memory.* Designs contain read-only memory (ROM) and either static or dynamic random-access memory (SRAM or DRAM). In addition, Micon also supports the use of cache memories when support chips exist in the microprocessor family, such as in the Intel 80386/80385 combination.
- *Peripherals.* All of the I/O operations are performed by dedicated parts such as serial and parallel I/O (SIO, PIO) devices. I/O devices can use standard interfaces such as RS-232Cs.
- *Bus interfaces.* The system supports a selection of standard bus interfaces (like the IBM-AT bus).
- *Support circuitry.* A set of circuitry provides support functions. This circuitry includes address decoders, wait-state generators, oscillators, and bus drivers, as well as specialized structures for improving system reliability, such as Hamming encoders/decoders.

The complexity of a Micon design roughly equals that of a small workstation.

As we intended Micon to provide a rapid prototyping facility, the overriding concern for the synthesizing system M1 was whether it would quickly generate working designs. Design costs—as measured by a variety of metrics including part cost, board area, and power dissipation—provide a framework for evaluating trade-offs. Since designs will not be produced in volume, a user is typically willing to lessen the cost constraints to obtain a design quickly. Cost optimization is not necessary. In general, designs produced by Micon compare favorably to human-generated designs. (The section on experiments and other experience provides more details.)

Figure 1 shows the Micon system. Note that we made a distinction between input from the user or the domain expert. The domain expert teaches M1 through the code generator (Cgen) how to design target structures and uses the data-entry tool to add new part models to the database. The user creates designs with M1 by exploiting the work of the domain expert. If the domain expert imparts sufficient knowledge to M1, the user need only be as knowledgeable as a novice hardware designer.

Input to M1 is a set of high-level specifications that describe the functionality required of the computer. For example, users can specify the type of microprocessor, amount and type of memory, and the number and type of I/O devices required. Users can also input an additional set of parameters describing multiple objective criteria, such as design constraints (board size, cost) and system-reliability requirements. M1 uses its knowledge of components and microprocessor system structures to develop a design that satisfies the requirements given to the system. During the design process, M1 interacts with the user to resolve trade-offs in the evolving design.

M1 is a rule-based system written in the OPS/83 development language. The rule base consists of knowledge about the components M1 can use in the design, microprocessor system-design techniques, and design-for-reliability techniques. Cgen updates the rule base with newly developed components and design techniques to produce competitive designs.

Automated Synthesis for Reliability (Assure[16]) is a submodule within M1 that analyzes and modifies designs for improved reliability. Assure's input consists of reliability criteria such as mean time to failure (MTTF) or mission time. Assure uses external reliability tools to analyze the quality of the evolving design. Designs that fail to meet the user-specified criteria are modified by a variety of techniques, ranging from simple IC packaging changes to the addition of entire structures (for example, error-correcting codes on memory). While Assure is closely tied to M1, it has a separate, distinct, problem-solving architecture.

Cgen[17] is the knowledge-acquisition tool for the M1 module. Typically, a rule base can only be updated by those intimately familiar with the program, which makes the acquisition of knowledge difficult. Cgen allows a hardware designer who is not familiar with M1's implementation details to add expertise to the rule base. Inputs to Cgen consist of schematic drawings and simple equations.

A suite of tools that perform the physical design function transforms the output of M1 into a complete computer.

Physical design consists of three steps. The first is placement and routing of the design for a board. A set of commercial physical design tools called Tango-PCB completes this task. The second step is fabricating the board. Two board-fabrication technologies are available: in-house wire-wrapping and printed circuit board (PCB) fabrication performed at a local PCB manufacturer. Populating the board with components and testing it comprise the final step.

The database serves as the central repository for the part models used by all modules in the system. We used a commercially available product, Informix, to build the database. All modules receive data from a common central database to ensure consistent information in the Micon system.

The Micon system is tightly integrated. All tools communicate via Unix interprocessor-communication links to the database. The database uses a server to establish communication links with each tool, or client. This scheme allows the database to run as a separate process. In addition to providing a great deal of flexibility—such as serving a large number of clients at the same time—it allows the dedication of a single workstation on the network as the database node. This arrangement centralizes the database management functions and eliminates many inconsistency problems. Figure 2 depicts the database server communicating with separate workstations across the network.

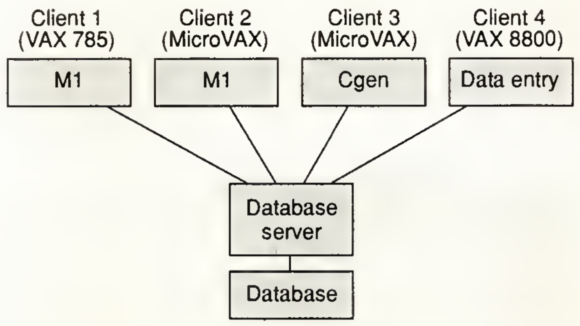


Figure 2. Database-server communications.

Experiments and other experience

We subjected Micon, as a working system, to a set of experiments to test its design capabilities. These experiments involved the following steps:

- teaching the system, through Cgen, to design with the set of microprocessors mentioned previously;
- exercising M1 to generate a set of designs; and
- passing an M1-generated design through the physical design and manufacturing processes.

Except for Birmingham, the designers who participated in the experiments had not implemented Micon.

The designers added data and knowledge concerning a large number of components to the system. The number of part models equaled 382. (See Birmingham and Siewiorek[17] for complete details.) The database contained every type of part necessary to build the computers, ranging from microprocessors to NAND gates and capacitors to RS-232C port connectors (DB-25s). M1's knowledge base contained over 919 design rules, all of which were created by Cgen. These rules described how to design with each of the parts in the database.

With its knowledge base and database, M1 can create an interesting variety of designs. Since knowledge concerning parts is individually acquired, Micon can configure unique combinations of parts into a design. The ultimate limitation on the designs comes from:

READER-SERVICE CARD

IEEE MICRO

October 1989 issue (card void after April 1990)

Name ____________________

Title ____________________

Company ____________________

Address ____________________

City ______________ State_____ ZIP ________

Country ______________ Phone (_____) ______________

Please send (Circle those you want)

201 Publications catalog
202 Membership information
203 Student membership information
204 Senior membership information
205 *IEEE Micro* subscription information

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	159	168	177	186
151	160	169	178	187
152	161	170	179	188
153	162	171	180	189
154	163	172	181	190
155	164	173	182	191
156	165	174	183	
157	166	175	184	
158	167	176	185	

Product Information 1
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	—
11	31	51	71	91	111	131	—
12	32	52	72	92	112	132	192
13	33	53	73	93	113	133	193
14	34	54	74	94	114	134	194
15	35	55	75	95	115	135	195
16	36	56	76	96	116	136	196
17	37	57	77	97	117	137	197
18	38	58	78	98	118	138	198
19	39	59	79	99	119	139	199
20	40	60	80	100	120	140	200

READER-SERVICE CARD

IEEE MICRO

October 1989 issue (card void after April 1990)

Name ____________________

Title ____________________

Company ____________________

Address ____________________

City ______________ State_____ ZIP ________

Country ______________ Phone (_____) ______________

Please send (Circle those you want)

201 Publications catalog
202 Membership information
203 Student membership information
204 Senior membership information
205 *IEEE Micro* subscription information

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	159	168	177	186
151	160	169	178	187
152	161	170	179	188
153	162	171	180	189
154	163	172	181	190
155	164	173	182	191
156	165	174	183	
157	166	175	184	
158	167	176	185	

Product Information 2
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	—
11	31	51	71	91	111	131	—
12	32	52	72	92	112	132	192
13	33	53	73	93	113	133	193
14	34	54	74	94	114	134	194
15	35	55	75	95	115	135	195
16	36	56	76	96	116	136	196
17	37	57	77	97	117	137	197
18	38	58	78	98	118	138	198
19	39	59	79	99	119	139	199
20	40	60	80	100	120	140	200

SUBSCRIPTION CARD

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

❑**YES**, sign me up!
If you are a member of the Computer Society or any other IEEE society, pay the member rate of only **$18** for a year's subscription (six issues).

Society: ______________ IEEE membership no.: ______________

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate ($9) for three bimonthly issues.

❑**YES**, sign me up!
If you are a member of ACM, ACS, BCS, IEE (UK), IEEE (but not a member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other qualified professional technical society, pay the sister-society rate of only **$33** for a year's subscription (six issues).

Organization: ______________ Membership no.: ______________

Full Signature *Date*

Name

Street

City

State/Country *ZIP/Postal Code*

❑ Payment enclosed

❑ Charge to ❑ Visa ❑ MasterCard ❑ American Express

Charge-card number

— — — — — — — — — — — — — — — —

Expiration date

— — — —
Month *Year*

Prices valid through 12/31/89
10/89 Micro

Change orders also taken by phone:
(714) 821-8380 8a.m. to 5 p.m. Pacific time
Circulation Dept.
PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: ______________________________

____

____

____

____

I disliked: ______________________________

____

____

I would like to see: ______________________________

____

____

Reviewers needed. *If interested, send professional data to Joe Hootman, EE Dept., University of North Dakota, PO Box 7165, Grand Forks, ND 58202.*

PO Box is for reader-service cards only.

PLACE POSTAGE HERE

IEEE Micro
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

Editorial comments

I liked: ______________________________

____

____

____

____

I disliked: ______________________________

____

____

I would like to see: ______________________________

____

____

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE POSTAGE HERE

IEEE Micro
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY
Circulation Dept.
PO Box 3014
Los Alamitos, CA 90720-9804
USA

Table 1.
Partial specifications of M1 designs.

	Design specifications					Results		
Set	Processor	SRAM (Kbytes)	ROM (Kbytes)	No. of PIOs	No. of SIOs	Part count	Actual runtime* (seconds)	Total runtime** (seconds)
1	MC68008	32	4	1	0	20	250	1,552
2	MC68008	60	4	1	0	20	257	1,591
3	MC68008	60	4	0	1	28	309	1,949
4	MC68008	60	4	1	1	31	354	4,000 †
5	MC68008	60	4	2	1	32	384	2,459
6	MC6809	32	4	1	0	17	227	1,991
7	MC6809	32	4	0	1	25	292	2,199
8	MC6809	32	4	1	1	27	331	4,067 †
9	MC6809	60	4	1	1	27	334	4,122 †
10	MC6809	60	4	2	1	28	361	2,277
11	80386	60	8	0	1	79	965	4,580

* On a MicroVAX II, Unix Version 4.3 BSD.
** Actual runtime plus time spent accessing shared database over the network. Database server runs on another MicroVAX II, Unix Version 4.3 BSD.
† Unusually large value for these items due to the sharing of the database with other programs running at the same time.

- the size of the memory space,
- the size of the I/O space (if applicable), and
- physical constraints (power, area, cost).

Table 1 shows the specifications used for a set of designs. These designs are a small sample of the M1's diversity and range. As we expected, the part count and runtimes increase for designs with more functionality (larger memory, more I/O ports). Notice that specification sets 1 and 2 (and also sets 8 and 9) are the same except for the amount of SRAM. To build a larger memory, M1 used a memory chip of larger density that resulted in the same part count. Also, set 5 just has one more PIO port than set 4 (as in sets 9 and 10). As the MC6821 chip used for the first PIO port actually has two ports, M1 uses the second port for set 5. Hence, the design for set 5 has only one more part than that for set 4. The additional part is a connector for the second port.

We have implemented the design M1 produced for the MC68008 using specification set 5 from Table 1. Analysis of the design showed the following.

- The design worked at the expected clock rates.
- Since the design was built almost entirely from VLSI components, it compared favorably with hand design in terms of part count, as expected.

We verified other M1-generated designs by manually converting the wire list to a schematic. We checked the correctness of the design when we created the schematic drawings. During this process, we found two categories of errors:

- *Missing parts.* Some schematic drawings mistakenly did not contain several support parts (for example, pull-up resistors and bypass capacitors).
- *Missing connections.* M1 did not make some necessary connections between components.

The errors all stemmed from mistakes in the schematics input to Cgen, which were propagated to M1's knowledge base. We corrected the schematics containing the errors, and Cgen generated new rules, which were inserted into M1's rule base. All designs became bug free.

Assure analyzed and modified an MC6809-based design produced by M1 for improved system reliability. Figure 3 on the next page graphs the steps taken by Assure in terms of the reliability goal, which was to maximize the MTTF.

Assure successively identifies the least reliable subsystem or part and applies a reliability-enhancement technique to it. Assure starts by applying simpler *fault-intolerant techniques* (like changing a package from plastic to ceramic or upgrading the quality of the package) until these result in marginal improvements. These improvements correspond to the flattening-out of the curve in Figure 3 (after five steps).

Assure next applies *fault-tolerant techniques* for the most failure-prone subsystem. Here, Assure added Single-Error Correcting/Double-Error Detecting (SEC/DED) structures to the SRAM and ROM array. Assure also attempted to triplicate the processor and the I/O peripherals but found that the fault tolerance did not compensate for the failure rate of additional support circuitry. Hence, Assure rejected those changes.

Since the experimental period, a user group has employed Micon extensively. The group has grown to include over a dozen persons, with participation from industry. In fact, we held two workshops with industrial participants. Micon functioned without flaw in this more realistic environment. (The industrial participants, while not malevolent, focused on testing the capabilities and identifying the flaws of the system.) Micon is currently being evaluated at several industrial sites.

The system has recently generated more sophisticated designs than those described here. In particular, the system generated an Intel 80386 design with cache, coprocessors, and AT-bus interface. The size of the system has grown to encompass over 500 part models and over 2,500 Cgen-acquired rules. We are beginning to add knowledge to M1's rule base for building microprocessors. We are also developing the capability to form a solid model of the design to evaluate its mechanical and thermal properties by using a mechanical modeling, air-flow analysis package.

Micon also functions as an element in a microprocessor interfacing course. As part of the laboratory requirement, students build knowledge bases of 80X86 family components and then generate and verify designs. Micon is a very powerful aid for teaching students good design methodology, a topic that is difficult to cover in the short period of one semester. Students in this class gain exposure to design issues not generally experienced until they begin industrial jobs.

The Micon system supports all aspects of computer design from logic synthesis to manufacture. The system can incorporate new technologies readily, which is a major strength that allows it to track current technology. Micon has proven its design capabilities through extensive use both inside and outside of the laboratory.

We continue to research Micon and are creating several aggressive designs with the system. In addition, we are porting Micon to domains outside of computer design.

Acknowledgments

This work was funded by the National Science Foundation grant DMC-8405136 to the Demeter Project and by the Engineering Design Research Center, Carnegie Mellon University, a National Science Foundation engineering research center supported by grant CDR-8522616.

The Micon system resulted from the efforts of many people. We thank Joe Najjar, Raj Merchia, Andrea Dusseau, Nino Vidovic, Ajai Kapoor, Audrey Brennan, and Patrick Edmond.

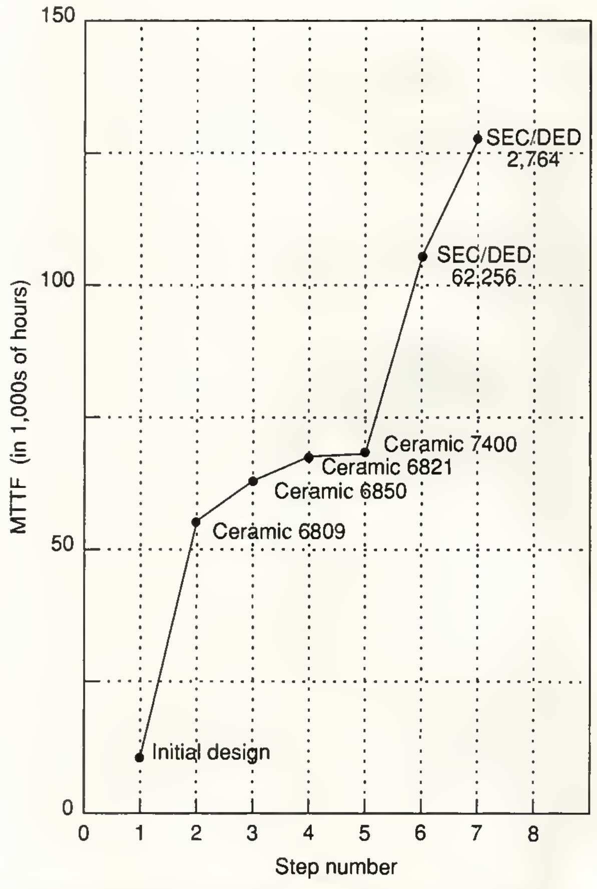


Figure 3. Assure's MTTF improvement plot for MC6809 design.

References

1. R. Harjani, R. Rutenbar, and L.R. Carley, "A Prototype for Knowledge-Based Analog Circuit Synthesis," *Proc. 24th Design Automation Conf.*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 42-49.
2. J.R. Dixon, "Artificial Intelligence and Design: A Mechanical Engineering View," *Proc. AAAI-86*, Morgan Kaufmann Publishers, Inc., San Mateo, Calif., 1986, pp. 872-877.
3. D.E. Thomas et al., "The System Architect's Workbench," *Proc. 25th Design Automation Conf.*, IEEE CS Press, 1988, pp. 337-343.
4. S.W. Director et al., "A Design Methodology and Computer Aids for Digital VLSI Systems," *IEEE Trans.*

Circuits and Systems, Vol. CAS-28, No. 7, 1981, pp. 634-645.

5. M.R. Barbacci, "Instruction Set Processor Specification (ISPS): The Notation and Its Applications," *IEEE Trans. Computer-Aided Design,* Vol. CAD-5, No. 4, 1981, pp. 24-40.
6. J.A. Darringer et al., "LSS: A System for Production Logic Synthesis," *IBM J. Research and Development,* Vol. 28, No. 5, Sept. 1984, pp. 537-545.
7. W. Birmingham and J. Kim, "DAS/Logic: A Rule-Based Logic Design Assistant," *2nd Conf. Artificial Intelligence Applications,* IEEE CS Press (microfiche), Dec. 1985, pp. 264-268.
8. D. Bobrow et al., "The Partitioning of Concerns in Digital System Design," Tech. Report VLSI-81-3, Xerox Palo Alto Research Centers, Palo Alto, Calif., 1981.
9. J. McDermott, "RI: A Rule-Based Configurer of Computer Systems," *Artificial Intelligence,* Vol. 19, No. 2, Sept. 1982, pp. 39-88.
10. T.J. Kowalski, *An Artificial Intelligence Approach to VLSI Design,* Kluwer Academic Press, Boston, 1985.
11. T.M. Mitchell, L.I. Steinberg, and J.S. Shulman, "A Knowledge-Based Approach to Design," *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 5, Sept. 1985, pp. 502-510.
12. V.E. Kelly, "The CRITTER System: Automated Critiquing of Digital Circuit Designs," *Proc. 21st Design Automation Conf.,* IEEE CS Press (microfiche), 1984, pp. 419-425.
13. W.P. Birmingham and D.P. Siewiorek, "MICON: A Knowledge-Based Single Board Computer Designer," *Proc. 21st Design Automation Conf.,* IEEE CS Press (microfiche), 1984, pp. 565-571.
14. A.P. Gupta, "A Hierarchical Problem Solving Architecture for Design Synthesis of Single Board Computers," Tech. Report CMUCAD-88-5, Carnegie Mellon University, ECE Dept., Pittsburgh, Feb. 1988.
15. W.P. Birmingham et al., "The Design of an Integrated Environment for the Automated Synthesis of Small Computer Systems," *Proc. 22nd Hawaii Int'l Conf. System Sciences* (Vol. 1: Architecture), IEEE CS Press, 1989, pp. 49-58.
16. A.A. Brennan, "Automatic Synthesis for Reliability," Tech. Report CMUCAD-88-2, Carnegie Mellon University, ECE Dept., Jan. 1988.
17. W.P. Birmingham and D.P. Siewiorek, "Capturing Designer Expertise—the CGEN System," *Proc. 26th Design Automation Conf.,* IEEE CS Press, 1989.

William P. Birmingham is an assistant professor in the Electrical Engineering and Computer Science Department at the University of Michigan, where he is also member of the Advanced Computer Architecture Laboratory. His background includes the design of several ICs and the development of AI-based CAD tools. Birmingham's current research interests include CAD, VLSI design, computer architecture, and AI.

Birmingham received the BS and PhD degrees in electrical engineering and the MS degree in computer engineering from Carnegie Mellon University. He is a member of the ACM, Sigma Xi, and the IEEE.

Anurag P. Gupta is a PhD candidate in the Department of Electrical and Computer Engineering at Carnegie Mellon University. His current research interests include CAD for synthesis of digital systems, computer architecture, and knowledge-based systems.

Gupta received the Bachelor of Technology degree in electrical engineering from the Indian Institute of Technology in Delhi and the MS degree in electrical engineering from Carnegie Mellon University.

Daniel P. Siewiorek is a professor in the School of Computer Science and the Carnegie Institute of Technology Department of Electrical and Computer Engineering at Carnegie Mellon University, where he helped to produce an operational 50-processor multiprocessor system in the Cm* project. He has contributed to the dependability design of 24 commercial computing systems, conducted research, served as a consultant, and published over 200 technical papers and five books, including *The Theory and Practice of Reliable System Design.* His current interests include computer architecture, reliability modeling, fault-tolerant computing, modular design, and design automation.

Siewiorek received the BS degree in electrical engineering from the University of Michigan, Ann Arbor, and the MS and PhD degrees in electrical engineering (with a minor in computer science) from Stanford University. He was elected an IEEE Fellow for contributions to the design of modular computer systems and has served as chair of the IEEE Technical Committee on Fault-Tolerant Computing. Siewiorek is a member of the ACM, Tau Beta Pi, Eta Kappa Nu, Sigma Xi, and the IEEE.

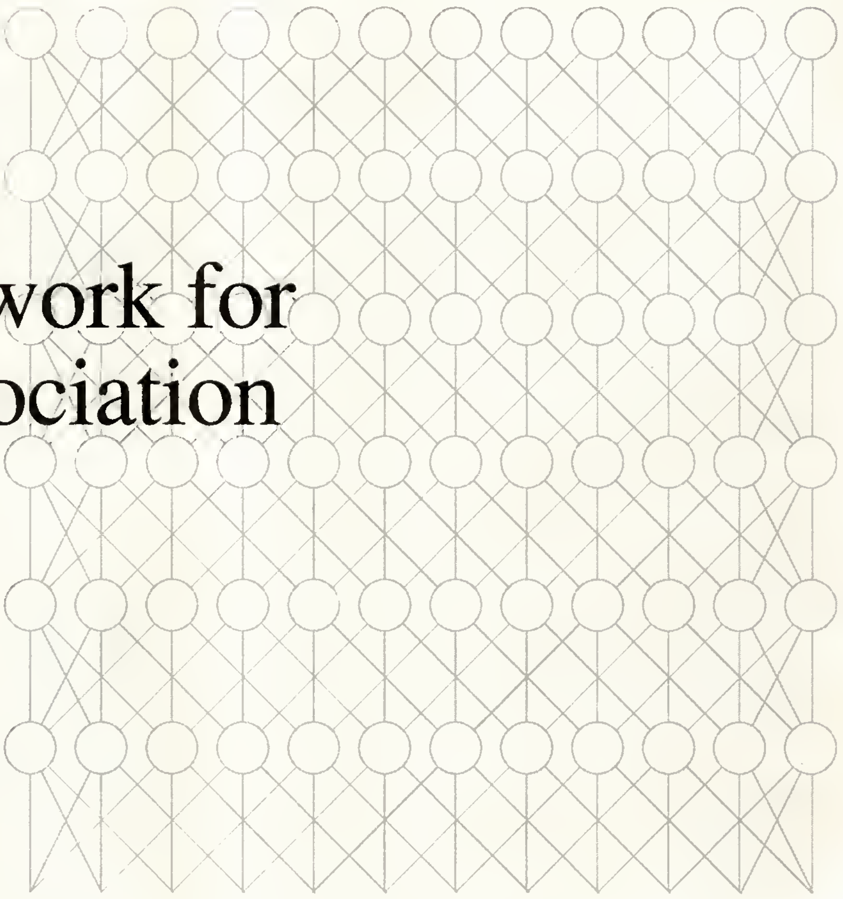
Questions regarding this article may be directed to William Birmingham, EECS Department, Advanced Computer Architecture Lab, University of Michigan, Ann Arbor, MI 48109-2110.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 168 **Medium** 169 **High** 170

SPECIAL FEATURE



A CMOS Neural Network for Pattern Association

A neural net replicated on a VLSI chip performs tasks formerly confined to software-only implementations.

Mark Walker
Paul Hasler
Lex Akers

Arizona State University

The need to develop systems that can replicate neural-like behavior in real time motivates the pursuit of hardware implementations of artificial neural network models. Neural-like behaviors include learning by example, the ability to adapt to different tasks, and fault tolerance.

The structure and function of biological systems inspire neural network designers. However, design rules and material characteristics that differ from these systems severely complicate the task of mapping the relevant characteristics of theoretical neural models into architectures constrained by IC technology. The most significant barrier is the high degree of connectivity required between the processing elements or neurons.

Neural network models in software generally consist of many very densely interconnected processing elements, each of which performs a simple computation in parallel with its neighbors. At present, very large scale integration, interconnection technology does not support the direct implementation of large-scale networks of this type. The area required to route connections and to contain the average length of interconnections increases at unacceptable rates as more processing elements are added.

Here we present an analog complementary metal-oxide semiconductor (CMOS) version of a model for pattern association, along with discussions of design philosophy, electrical results, and a chip architecture for a 512-element, feed-forward IC.

An earlier version of this article appeared in the *Proceedings of the 8th Annual International Phoenix Conference on Computers and Communications*, 1989.

0272-1732/89/1000-0068$01.00 © 1989 IEEE

Hardware implementations

Researchers have proposed—and actually fabricated—a number of hardware realizations of neural networks. A majority of the designs are digital and employ either standard digital VLSI technology[1] or more specialized architectures such as digital signal processors[2] or systolic arrays.[3] Researchers have also proposed or fabricated analog[4] and hybrid analog/digital[5-8] designs.

Electronic neural networks also differ in the manner in which the weights are stored (see the processing-element design section). Some designs store weights as binary vectors,[7,8] while others represent weights in the form of dynamic charges stored on capacitors.[5,6] Still others represent weights as static charges stored in electrically erasable programmable read-only memory devices (EEPROMs).[9] Current designs employ either limited interconnections[5] or some means of achieving full interconnections.[10]

Comparing the neural models implemented by different architectures provides additional insight. Neural network models generally fall into four main classifications according to which neural-like behaviors they replicate. These categories are

- autoassociation,
- input classification,
- regularity detection, and
- pattern association.[11]

Several models can perform tasks that span several categories. Many of the neural architectures are designed to perform the task of autoassociation, or associative memory. Autoassociation typically requires a fully connected, single-layer (each processing element or node connected to all others) network with symmetric, bidirectional transmittances between processing elements.[12] The magnitudes of the transmittances are variable parameters of the network and can be adapted with a learning paradigm. This process gives rise to a distributed storage of previously trained responses.[13] The system can require inputs in the form of binary vectors. The entire network makes excursions through state space until it reaches a stable state. The final state is usually a previously stored vector that represents the pattern that most closely matches the original input.

Perhaps the best known hardware realization of an associative network architecture is the one designed by the AT&T Bell Laboratories' Holmdel group.[4] This chip is an analog implementation consisting of summing amplifiers globally connected by thin-film resistors. Some researchers have suggested networks of this type for application in systems where "noisy" inputs must associate with codes representing a finite set of stored symbols.

Hardware implementations of networks that can accomplish arbitrary pattern association are less common, since they generally require layered approaches. Also, the transmittances require more complex learning algorithms to achieve adaptation.

Neural networks could serve as very powerful computational paradigms for hardware realization.

Probably the most well-known model in the layered network category is the *perceptron*.[14] In many applications, the perceptron is adapted with an algorithm that is a modified version of least-mean-square learning.[15] This algorithm is known as the Delta Rule or, more commonly, back propagation.[16] Using the Delta Rule, the model trains layered networks by repeatedly presenting pairs of example vectors at the input and output layers. It then incrementally adjusts the internal network transmittances so that when the first vector in each pair is presented as input, the second appears as output.

When the model adapts the magnitudes of the intralayer transmittances to produce the set of desired outputs for the applied, trained inputs, the network develops a sensitivity to the general, invariant properties or "rules" relating input to output. Generalization can occur when a learning algorithm endows a network with a general sensitivity to these properties. This process allows correct inference of outputs not included in the training set. Neural networks therefore could serve as very powerful computational paradigms for hardware realization. They can theoretically adapt arbitrarily complex I/O functions through training.

Unlike one-layer networks, the multilayered perceptron employs feedback only during training. This approach allows it to operate in an essentially feedforward, deterministic manner after training is completed. This model is therefore well suited for discrete-time operation when it is embedded in larger digital systems. It serves as the basis for the limited-interconnection, CMOS network we discuss.

Effect of limited interconnections

We have simulated the limited-interconnection network to determine the performance limitations imposed by limited fan-in at each node as well as the effect of using a fixed topology architecture for different tasks. We trained the network in separate runs to per-

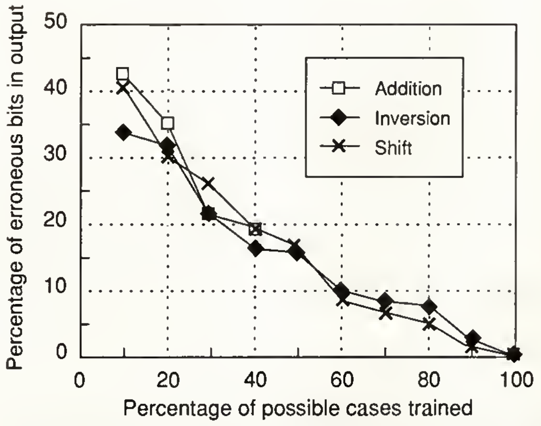


Figure 1. Network response error as a function of training.

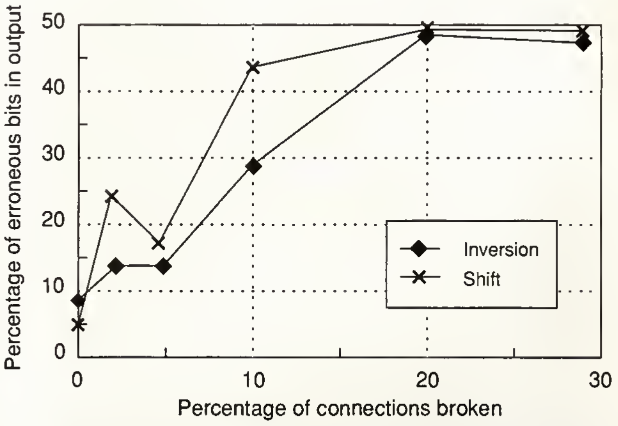


Figure 2. Network response error as a function of failed connections (80 percent of all possible 8-bit example pairs were trained).

form three different operations on an 8-bit, digital input vector using the Delta Rule. The tasks implemented included inversion, shifting to the right, and unsigned, one's-complement addition. Well-posed problems of this type are usually poor candidates for neural network implementation. They are, however, good worst case algorithms for evaluating this architecture, since they require a one-to-one mapping (eight-dimensional input space to eight-dimensional output space). In addition, the shifting and inversion input spaces are linearly separable along the desired hyperplanes.

The total number of connections within an adaptive filter determines the degrees of freedom available to the network in forming hypersurfaces that correctly divide the input space.[17] Consequently, a loose design constraint for limited-interconnection, multilayered neural networks is that they must contain at least as many connections as the smallest, fully connected structure that implements the same function. A fully connected, two-layer perceptron (one layer of "hidden" units) is generally considered to be the smallest structure that can generate arbitrarily complex mappings.[18] In these tests, a two-layer network with 10 hidden units was the smallest fully connected structure that implemented all three tasks with an acceptable amount of error.

Because double-metal CMOS design rules limit the generation of dense interconnection patterns, we selected a fan-in of four for each processing element. A higher number of possible inputs at each cell would require channel-routing schemes between adjacent layers. The final design of the limited-interconnection structure had approximately the same total number of connections as the fully connected structure. It maintained the required fan-in as a four-layer network with three hidden layers of 10 units each. More precise design methods for feed-forward networks that form distributed internal representations are not yet available. Designers usually determine empirically the number of internal processing elements required in actual applications of perceptron networks.

Our first set of results demonstrates the ability of the limited-interconnection structure to generalize untrained outputs from a limited set of examples. Figure 1 shows the response error as a function of training. Even though the three tasks for which the network was trained have very different governing algorithms, each network inferred untrained output in an almost identically linear fashion.

The second simulation demonstrates the relative fault tolerance of the limited-interconnection architecture. In this experiment, an increasing number of randomly selected connections within the network were selected and broken (transmittance set to zero). Figure 2 demonstrates that the networks performing inversion and shifting continued to function at the same level in spite of a small number of broken connections. The actual amount of error introduced in the output, however, depended on the particular connection broken and the task learned. The Delta Rule randomly determined the relative importance of each connection in generating the desired mapping. Also, since we are employing a fixed network topology for different tasks, the amount of redundancy within the network depends on the function implemented. For these reasons, many of the weights may contribute little to the desired behavior of the network and eliminating them does not significantly degrade performance.

The network performing addition, however, did not survive the loss of a few connections. The number of connections provided apparently did not allow for redundancy. Greater fault tolerance might have occurred with a larger network.

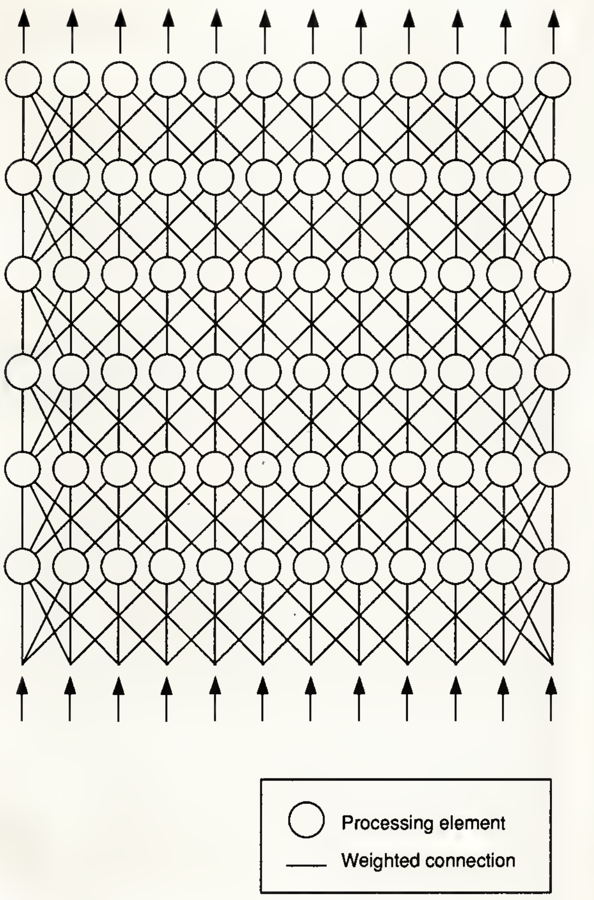


Figure 3. Network interconnection scheme.

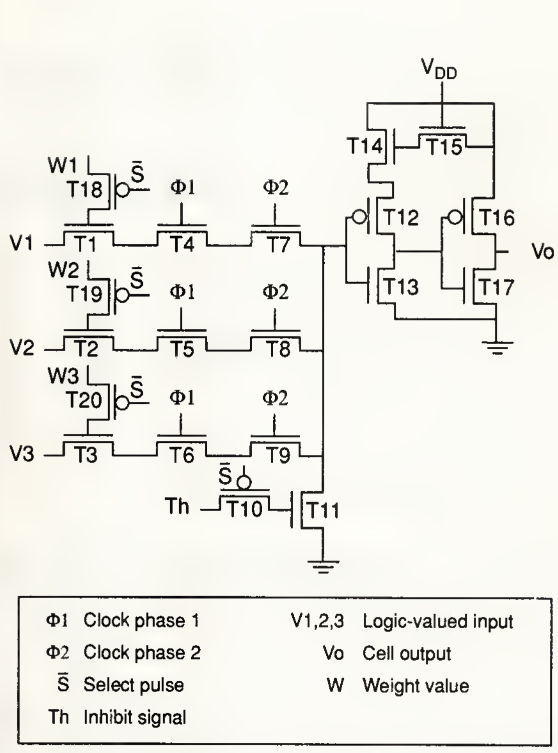


Figure 4. Analog synthetic neural cell.

Network design

The number of devices on a die have increased in a exponential fashion over the last 28 years. Regretfully, the number of interconnection layers has not increased at this rate. This state of affairs causes serious restrictions on information flow in VLSI and ultra LSI systems.[19] It also has important consequences for hardware implementations of synthetic neural networks. We must design cells and networks that do not require high interconnectivity when they are scaled to large numbers of processing elements. As indicated, we can eliminate global interconnections between layers by substituting local interconnections and additional layers. This is an excellent trade-off for VLSI technology because we can rapidly design and implement large systems by replicating the cells in both row and column directions. We need only modify the select logic on the die. Figure 3 shows the symbolic layout of a 12-input network implemented with this approach.

Processing-element design

In most layered perceptron models, each processing element operates by taking a weighted sum of its inputs. The sum then passes through a threshold function or a high-gain, sigmoidal nonlinearity. The compactness of the cell design presented here is due in large part to the use of analog transmittances and charge summation in the implementation of the input weighting function.

Figure 4 is the circuit diagram of the limited-interconnection analog neural cell. The operation of the cell is as follows. Weights are stored dynamically on the gates of transistors T1, T2, and T3. Since the inputs to each device will be 0 or 5 volts, a voltage on the gate of any of the N-channel MOS input transistors charges the output to a degraded value (by a threshold voltage). This value equals the transmittance of the input transistor multiplied by the logic-valued input (0 or 1). Since a transmittance smaller than a threshold cuts off the input signal, a P-channel MOS pass transistor can directly load a transmittance value onto the gate of the device. This type of transistor passes everything above the device threshold level.

For inputs of 5 volts, the drain parasitic capacitors of transistors T7, T8, and T9 are charged by current that is flowing through the pass transistors T4, T5, and T6. The voltage equals the weights minus the device threshold voltage. For inputs of 0 volts, the pass transistors

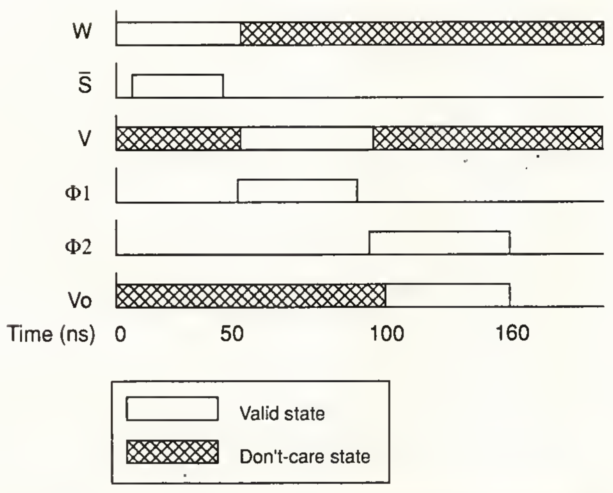


Figure 5. Simulation timing diagram for an analog neural cell.

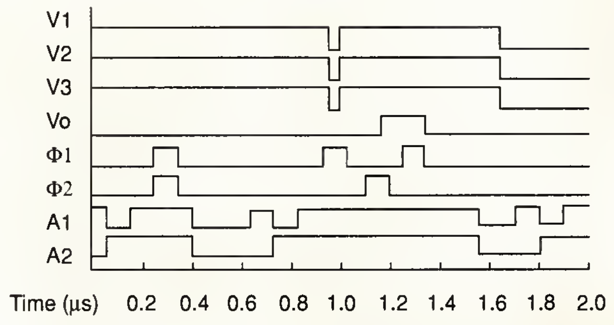


Figure 6. Experimentally obtained cell timing diagram—voltage vs. time.

allow the capacitors to discharge. Although exact multiplication of the input and the transmittance does not occur, modifying the training algorithm can compensate for this behavior.

Figure 5 is the timing diagram for a single synaptic branch of a three-branch neural cell. Once the storage capacitor in each branch is charged, clock phase 1 (Φ1) turns off to isolate the signal from the input (V). Turning on clock phase 2 (Φ2) allows the analog summation of the weighted input signals, which is compared to the logical threshold of the first inverter (which comprises T12 and T13 in Figure 4). The first inverter should be of minimum size to optimize charge transfer. Transistors T14 and T15 on top of the PMOS pull-up device establish the logical threshold and hence allow the neural threshold to be set at a voltage lower than $0.5V_{DD}$. The output inverter restores the output voltage level Vo and drives the next stage. Since this circuit uses only positive weights, a shunting transistor T11 provides inhibition.

Test chips were fabricated in 3-micrometer, p-well CMOS. Figure 6 shows the results of cell electrical tests using an HP16500 testing device. The three inputs are labeled V1, V2, and V3, while the output is Vo. The clocks are Φ1 and Φ2; the address lines are A1 and A2. For this example, the three inputs and weights were held high. The output becomes high upon completion of the summing of the inputs at the end of Φ2.

System operation

A system formed by replicating the analog cell operates in the following manner. Sharing the inputs, weight lines, and output lines keeps the number of I/O pins at reasonable numbers as the system expands. For the implementation of 512 neurons in a network 32 elements wide by 16 elements long, 32 lines are used for the weights and I/O. The transmittances are multiplexed onto a single line running to each cell. When a row is selected for a write, the system writes the weights for each cell. This procedure continues for four cycles, after which all weights for the row addressed have been written. The next row can then be selected for transmittance loading. The entire network can be loaded in approximately 10 microseconds, which is an order of magnitude faster than the discharge time constant for an individual gate. Figure 7 illustrates the floor plan of the chip.

One can pipeline the operation of the network for maximum throughput if careful attention is paid to the clearing of data that must follow every row of input data. A potential danger with storage capacitors is that a transmittance value below the device threshold with a low input does not allow the capacitor to discharge. We have eliminated this problem by pipelining a wave of zeros through the network. In addition, the Φ1 and Φ2 lines can be swapped every other row to synchronize pipeline operation. This procedure facilitates the multiplexing and encoding of the input data.

We have designed, simulated, and tested an adaptive, CMOS neural network for pattern association. We employed limited interconnection between adjacent layers of processing elements to reduce connection densities to VLSI-implementable levels in this synthetic neural IC design.

Simulations demonstrate that substituting global interconnection with limited interconnection and additional layers does not significantly degrade desired neural-like behaviors. We have developed a compact analog neural cell that functions with a dynamic RAM fabrication line and that drives a style of architecture that is fully compatible with LSI technology.

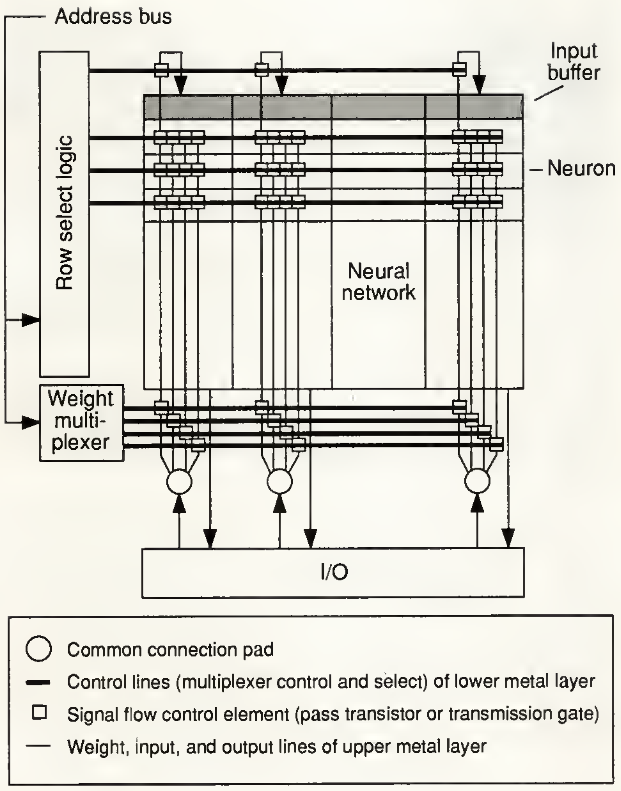

Address bus
Input buffer
Neuron
Row select logic
Neural network
Weight multi-plexer
I/O
○ Common connection pad
▬ Control lines (multiplexer control and select) of lower metal layer
□ Signal flow control element (pass transistor or transmission gate)
— Weight, input, and output lines of upper metal layer

Figure 7. Circuit floor plan.

Acknowledgments

We gratefully acknowledge the assistance of the neural network research group at Arizona State University: Larry Clark, Jennifer Wang, W. Fu, Ataru Shimodaire, C.C. Goh, Siamack Haghighi, Al Afghan, and Arun Rao.

References

1. A.P. Thakoor et al., "Electronic Implementations of Neural Networks," *Applied Optics*, Vol. 26, No. 23, Dec. 1987, pp. 5085-5092.
2. P.A. Penz and R. Wiggins, "Digital Signal Processor Accelerators for Neural Network Simulators," *Neural Networks for Computing*, J.S. Denker, ed., AIP Press, New York, 1986.
3. S.Y. Kung and J.N. Hwang, "Parallel Architectures for Artificial Neural Nets," *Proc. IEEE Int'l Conf. Neural Networks*, Vol. II, 1988, pp. 165-172.
4. H.P. Graf, L.D. Jackel, and W.E. Hubbard, "VLSI Implementation of a Neural Network Model," *Computer*, Vol. 21, No. 3, Mar. 1988, pp. 41-51.
5. L.A. Akers and M.R. Walker, "A Limited-Interconnect Synthetic Neural IC," *Proc. IEEE Int'l Conf. Neural Networks*, Vol. II, IEEE Press, Piscataway, N.J., 1988, pp. 151-158.
6. J.P. Sage, K. Thompson, and R.S. Withers, "An Artificial Neural Network Integrated Circuit Based on MNOS/CCD Principles," *Neural Networks for Computing*, John Denker, ed., AIP Press, 1986.
7. J. Alspector, B. Gupta, and R.B. Allen, "Performance of a Stochastic Learning Microchip," *Advances in Neural Information Processing Systems*, D.S. Touretzky, ed., Morgan-Kaufmann Publishers, Inc., San Mateo, Calif., 1989.
8. A.F. Murray, A. Hamilton, and L. Tarassenko, "Programmable Analog Pulse Firing Neural Networks," *Advances in Neural Information Processing Systems*, D.S. Touretzky, ed., Morgan-Kaufmann Publishers, Inc., 1989.
9. M. Holler et al., "An Electrically Trainable Artificial Neural Network (ETANN) with 10240 'Floating Gate' Synapses," *Proc. Int'l Joint Conf. Neural Networks*, Vol. II, 1989, pp. 191-196.
10. J. Bailey and D. Hammerstrom, "Why VLSI Implementations of Associative VLCNs Require Connection Multiplexing," *Proc. IEEE Int'l Conf. Neural Networks*, Vol. II, 1988, pp. 173-180.
11. D.E. Rumelhart and D. Zipser, "Competitive Learning," *Parallel Distributed Processing, Vol. 1: Foundations*, D.E. Rumelhart and J.L. McClelland, eds., MIT Press, Cambridge, Mass., 1986.
12. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat'l Academy of Sciences*, Vol. 79, Apr. 1982, pp. 2554-2558.
13. G.E. Hinton, J.L. McClelland, and D.E. Rumelhart, "Distributed Representations," *Parallel Distributed Processing, Vol. 1: Foundations*, D.E. Rumelhart and J.L. McClelland, eds., MIT Press, 1986.
14. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington, D.C., 1962.
15. B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," *1960 Wescon Convention Record*, Part IV, 1960, Institute of Radio Engineers, New York, pp. 96-104.
16. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing, Vol. 1: Foundations*, D.E. Rumelhart and J.L. McClelland, eds., MIT Press, 1986.
17. B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.
18. W.Y. Huang and R.P. Lippmann, "Neural Net and Conventional Classifiers," *Neural Information Processing Systems*, D. Anderson, ed., AIP Press, 1987.
19. D.K. Ferry, L.A. Akers, and E. Greeneich, *Ultra Large Scale Integrated Microelectronics*, Prentice-Hall, Inc., 1988.

Mark Walker received his BSEE and MSEE degrees from Arizona State University in Tempe, where he is currently a PhD candidate in the field of electrical engineering. His research interests include computational neurobiology, synthetic neural network implementations, and artificial vision.

Walker is a member of Tau Beta Pi, Eta Kappa Nu, and a student member of the IEEE and the IEEE Computer Society.

Paul Hasler is currently pursuing the BSEE and MSEE degrees at Arizona State University. His research interests include analog VLSI design, synthetic neural network implementations, and optical circuit design. Hasler was a finalist in the 1988 IEEE Region 6 student paper contest.

Hasler is a member of Tau Beta Pi and Eta Kappa Nu.

Lex Akers is a professor in the Department of Electrical and Computer Engineering at Arizona State University, where he is currently working on neural network and submicrometer CMOS development. Akers was a National Aeronautics and Space Administration fellow at the California Institute of Technology Jet Propulsion Laboratory. He has held positions in the Microelectronics Research and Development Center at Rockwell International and in the Advanced Products and the Semiconductor Research and Development Laboratories at Motorola.

Akers received the BSEE, MSEE, and PhD degrees from Texas Tech University. He is a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, and is past chair of the Phoenix chapter of IEEE Electron Devices Society.

Questions regarding this article may be directed to Mark Walker, Arizona State University, College of Engineering and Applied Sciences, CT1 1001, Center for Solid State Electronics Research—ESS, Tempe, AZ 85287-6206.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 162 **Medium** 163 **High** 164

Call for Papers

***IEEE Micro* seeks manuscripts for general-interest issues in 1990.**

Submit manuscripts to:
Joe Hootman, Editor-in-Chief, EE Dept., University of North Dakota, PO Box 7165, Grand Forks, ND 58202, phone (701) 777-4331.

Topics of particular interest include

- ❑ neural networks
- ❑ artificial intelligence
- ❑ special-purpose computers
- ❑ optical computers and interfaces
- ❑ workstations
- ❑ use of microprocessors in parallel computers
- ❑ VHDL design
- ❑ silicon compilation
- ❑ biological computing
- ❑ and tutorials on all micro-related topics.

Micro Law

continued from p. 9

to criminal penalties. Not only is such copyright infringement a crime in itself, but it may also lay the basis for a conviction for mail or wire fraud and for violation of the Racketeering Influenced and Corrupt Organization (RICO) Act.

Two basic elements must be present to prove copyright infringement. The alleged pirate must have had an opportunity to copy the copyrighted work (had access to it). The allegedly piratical version must be "substantially similar" to the original, copyrighted version. Determining substantial similarity is a well-defined process in the case of novels, paintings, and songs. However, it is inevitably unpredictable because it calls for making essentially subjective judgments. In the case of computer programs the concept of substantial similarity is not yet well defined. Accordingly, determining whether program A is substantially similar to program B must be done without definite standards, in the current state of copyright case law.

Protecting computer programs. A third, more subtle element of proof of copyright infringement is that the material the alleged pirate took from the copyright owner's work was protected by the copyright. According to settled copyright doctrine, the copyright laws protect only the *expressions* of ideas, but do not protect the underlying *ideas* themselves. The copyright in a book about making microcomputers will keep others from copying and selling the book. But it will not keep them from manufacturing microcomputers in accordance with the teachings of the book. Preventing this may be done, if at all, only under the patent laws. Unlike copyrights, of course, patents issue only after an expert examination of the novelty and technical merit of the invention.

The words of a book plus some limited penumbra of organization and style about them comprise authors' expression of their ideas. The substantive content, however, is unprotectible idea. When the expression is functional, and few or no other modes of expression will accomplish the same purpose, the expression is said to "merge" with the idea and thus become unprotectible.

For example, paraphrases of the words of a novel may well be copyright infringements. The words used to state instructions for playing a game or for using a computer program may be less open to idiosyncratic variation than those of a novel and thus may be considered to merge with the idea. Hence, a paraphrase of them may well not be a copyright infringement. To copy or paraphrase such words is to take idea rather than expression, if the expression in such words has merged with the idea. Taking ideas is not copyright infringement.

Similarly, spreadsheet formats, Nyquist charts, Bode charts, and other forms or diagrams developed to carry out the teachings of a particular new method of doing something (for example, of double-entry bookkeeping or devising electronic systems) may be first disclosed in a copyrighted book explaining the new method. Nevertheless, if it is necessary to use forms or diagrams, or ones similar to them, to practice the new method, the forms and diagrams are considered as "necessary incidents" to the new method. They fall into the public domain along with the new method. (The new method would not necessarily fall into the public domain if it were the kind of thing that it is possible to patent, and the author patents it.)

The copyright status of screen displays for computer programs remains confused.

Although these principles can be applied to the traditional subject matter of copyright without undue difficulty, they do not always readily fit computer programs. Courts disagree about what aspects of a program should be considered functional or utilitarian. For example, one court has held that commercial need or desire for compatibility is not an aspect of function. Hence, the court refused to excuse copying lines of basic input-output system (BIOS) code as the taking of idea rather than of expression. The court found copyright infringement even though the copied BIOS code was allegedly needed to make computers and computer programs compatible with an established, installed base of computers and software written for them.

But another court has held substantially the opposite. It ruled that the right to engage in second-sourcing a chip (selling a completely substitutable chip) legitimized some copying. In that case, the accused copyright infringer had the right to second-source a microprocessor chip and make a fully compatible second chip. The court held that this meant that it was also legitimate to write the microcode for the second microprocessor chip in a way that replicated some aspects of the microcode of the original microprocessor (the lines of code of some microcode instructions).

Courts also disagree about whether the copyright in a computer program extends to anything more than the particular lines of code or a close paraphrase of them. Most programmers will agree that translating a program from Basic to Cobol would be sufficiently easy that marketing the resulting program should be a copyright infringement. But when the "translation," if it may even be called that, is from Basic to APL or to some other kind of language very different from Basic, programmers tend to disagree. They disagree over whether marketing the resulting computer program should be regarded as a copyright infringement. Nevertheless, that even some translations of code are considered infringements indicates that the scope of a copyright in a computer program is wider than merely the specific lines of code in it.

Some courts have said that copyright protections apply much more broadly than just to the lines of code of a computer program and to translations and close paraphrases of them. These courts hold that the protections extend to the "sequence," "structure," and "organization" of computer programs, as well. Such courts do not rigorously define sequence, structure, and organization. But their meanings seem to include such things as the way subroutines and other modules are partitioned from the rest of the program and the selection of data fields.[1]

One recent decision has protected the "sequence and flow" of screen displays as part of the protection for a computer program.[2] Moreover, while individual algorithms are considered ideas, it has been argued that the particular choice and selection of algorithms to implement in a computer program is part of structure and expression.

Even under the structure dispensation, however, the structures of two computer programs (using that term in the senses indicated above) may be quite dissimilar and still generate the same visual display. The only way for the "structure" of a computer program to embrace the screen displays associated with the computer program would seem to be by outright judicial fiat. That would be for the court to define the structure of computer programs as including their visual displays. In other words, the visual output that results from operation of a computer program would be the same thing as the internal organization of the computer program.

One court went so far as to suggest this. But a later decision by another court stated that this was a mistaken idea and that the first court had "overextended the scope of copyright protection applicable to screen displays." (However, the second court then held that the sequence and flow of successive screens was comprehended in the copyright on the computer program generating the screen displays.) Such a fiat simply disregards the fact that there is no particular correlation between the internal organization of display-generating code and the visual appearance of the display.

Case law on screen displays. Given this general state of uncertainty in the computer copyright field, the copyright status of screen displays for computer programs remains confused. In one recent case, the Softklone case whose screen display I discussed earlier in this series,[3] the court held that a menu screen display must be considered a separate work from the underlying code of the associated computer program. Therefore, it did not matter that the code of the defendant's computer program was not substantially similar to the code of the plaintiff's program. The court found copyright infringement because the two parties' menu displays looked substantially similar and the plaintiff had properly registered its display separately from the code. Another recent decision endorses the reasoning of Softklone on this point.

In principle, this is the most logical view to take. Moreover, it is consistent with the view that courts have taken of video-game copyrights. As indicated earlier, video-game screen displays are similar to those associated with other types of computer programs. Their treatment under the copyright laws may therefore suggest how screen displays, in general, should be regarded for copyright purposes. Whether video games could be protected by the copyright laws was at first uncertain. Initially, it seemed that only the codes of the computer programs used with the video games were protectible, although no decision expressly so held. After a time, the courts came to the view that the images used in the video games were protectible, apart from the underlying codes, as "audiovisual works."

The courts in addressing video-game copyrights noted that many different codes could be written to generate the same video-game imagery. They may also have been moved by another fact. One group of persons created the imagery and plot of commercially successful video games (such as Donkey Kong), and a different group wrote the necessary code. (That has been true for screen displays of computer programs such as 1-2-3 also.) In any case, the courts were prepared to find copyright infringement of audiovisual works when visual appearances were similar and codes were dissimilar, on the ground that the two things were separate works of authorship, each of which was entitled to independent protection in its own right.

The considerable difference in approach that these decisions take indicates the uncertain state of the law as to screen displays.

Finally, in a case involving a variation on the Pac-Man video game, the court found that the defendant's code was approximately 90 percent bit for bit the same as the plaintiff's code. Therefore, the defendant infringed the copyright in the code. But the defendant's rather slight changes in the code had resulted in visual imagery that did not resemble that of Pac-Man. Hence, the court found no infringement of the copyright that existed in the audiovisual work, but there was infringement of the copyright in the code. Thus, the video-game decisions recognize that screen displays and the code associated with them are distinct from one another. Rather than a one-to-one relationship, they have a many-to-many relationship.

The Softklone decision treating computer program screen displays as wholly distinct from code thus appears to be consistent with the reasoning used in the video-game cases. Other court decisions, however, support the view that protection of screen displays may be based on the copyright in the computer program considered as a whole or on the copyright in its code.

In one 1986 case the court protected the screen displays generated by the program as audiovisual works. It is unclear from the opinion whether the screens had been registered with the Copyright Office separately from the underlying code (which apparently differed considerably between the two programs). If they were, again the result would appear sound in principle and consistent with the video-game precedents. Some of the court's language, however, suggests a less sound rule. The court may well have considered the copyright in the code of the plaintiff's computer program to extend to the screen displays of the computer program. Then, that similarity between the two parties' screen displays in some unexplained way was supposed to make their intrinsically different computer program codes substantially similar. Two later decisions rejected this approach as erroneous.

In one extreme 1985 decision, the trial court stressed the close similarity of the parties' screen displays and the fact that the two computer programs produced results that were indistinguishable to naive users. Therefore, even though the trial court did not find the codes (which were written in different programming languages) to be similar, even on a translation basis, the court nonetheless found copyright infringement because the screen displays looked similar. On appeal of this decision, however, the court of appeals proved unwilling to place much weight on the similarity of the screen displays. It found the computer programs substantially similar and thus infringing for other reasons.[1,4]

The considerable difference in approach that these decisions take indicates the uncertain state of the law as to screen displays. Clearly, the difficulty of the subject matter and the tribunals' lack of technological education are factors in the disorder of the case law.

Another factor weighs heavily. Courts in some of these cases may be moved by the semblance of unfair competition and misappropriation. The defendant may appear clearly to be appropriating the visual effects that the plaintiff created, even if not the plaintiff's coding efforts. This view often imparts an irresistible impulse to a court to give the parties their just deserts, by protecting the first comer against the second comer's competitive imitation. Kaplan of Harvard referred to this as the Manichaean heresy of copyright law. Desire to achieve equity may overcome traditional copyright doctrine and its limitations on the scope of legal protection. The desire may run roughshod over any balanced view of the need of workers in the field to have a technical repertory or toolkit. Thus, perceived equity may conflict with the requirements of software progress. At the very least, a seemingly equitable result may be realized via a shoddy analysis.

The Copyright Office

An important variable in predicting the scope of copyright protection is the position that the US Copyright Office takes in the exercise of its registration function and in its rulemaking capacity. The Office does not promulgate the copyright law; that responsibility belongs to Congress. Congress legislates, and the courts interpret the legislation. The task of the Copyright Office is to examine applications for registration of copyright to determine whether the applications and the works to be registered meet the requirements of the copyright laws. In addition, the Office is authorized to issue regulations clarifying technical aspects of copyright law, insofar as they affect the registration of copyrights.

Despite the limited scope of the Office's authority, the courts and Congress give considerable deference to the views and administrative actions of the Office in carrying out its responsibilities under the copyright laws. Moreover, the registration of a claim of copyright is presumptive evidence that the copyright is valid, and that the applicant's statements in the application form are correct. Accordingly, the Office's determination that screen displays should be considered separate works from the underlying code generating them, and should therefore be separately registered, will probably have a significant impact on future court decisions in this field.

Before Softklone. The Copyright Office began with no position at all on whether screen displays for computer programs were separately registrable, presumably because it did not recognize that an issue existed. It therefore registered some screen displays, as such, apparently inadvertently. However, the registration was consistent with its and the courts' position that video-game displays were independently copyrightable.

The Office then adopted a position against separate registration of screen displays. When Lotus sought to register the screens for its 1-2-3 spreadsheet program, as a preliminary step to suing two competitors, the Office refused to register the screens as such. In a January 1987 letter it stated:

> It is the position of the Copyright Office that textual screen displays embodied within the computer program that generates them are covered by the registration for the programs, without either the need or justification for separate registration for the displays. Because the displays are considered to be an integral part of the program, the authorship in the displays appears to be the same as that contained in the program. Moreover, the Copyright Office would not register a claim in the format or layout.

The public and the courts simply have to make their own ways in the world.

After its refusal to register the screen displays of 1-2-3, the Copyright Office refused generally to permit menus and similar nonpictorial screen displays of computer programs to be registered separately. It would not accept deposits of printouts or photographs of screens, or other material identifying them. The Office's policy was that the registration of the source or object code of a computer program inherently included whatever copyrightable content existed in the screens displayed when the program was executed.

The Softklone court's rejection of that theory, however, caused the Office to feel that it needed to reevaluate its policy. It solicited public comment on whether it should change its policy, and it held public hearings in September 1987 at which various industry members and groups stated their positions. At these hearings, industry spokesmen and Computer Society[5] representatives urged various positions on the Copyright Office. These positions ran the whole spectrum of possibilities on whether screen displays should be considered separate works from computer programs associated with them, and thus separately copyrightable, registrable, and protectible.

After Softklone. After being exposed to all of these views, the Copyright Office decided to revert to its pre-Softklone position against separate registration of screen displays, with several minor exceptions. One was that the Office would allow the identifying material submitted with applications for registration of computer programs to include pictures of, or other material identifying, the screen displays associated with the programs. Another change was that the Office would allow the work to be identified in the application form either as "computer program" or "computer program code and screen displays." In either case, the Office would consider the registration to cover the entire copyrightable material in the code and screen displays.

The Copyright Office did not greatly elaborate on its above-quoted 1-2-3 statement in its June 1988 announcement of policy on copyrighting screen displays separately. It preferred to stick to a terse, conclusory assertion:

> The Office finds that in the interest of a clear, consistent public record, our registration practices should discourage piecemeal registration of parts of works.
>
> ... Copyright claimants are able to register all copyrightable elements contained in their work with a single application and fee [$10].
>
> The public is benefited through the maintenance of a clear, accurate, easily understandable public record. Permitting multiple registrations of parts of works would increase confusion among those attempting to use the records of the Copyright Office. Subdividing claims [of copyright ownership] might also result in multiple infringement actions and multiple claims for statutory damages, based on separate registrations.
>
> The Copyright Office benefits by having a simplified administrative analysis.

The Office's June 1988 policy statement does not expressly address the Softklone court's statement rejecting the policy that the Office adopted. The Office acknowledged in its statement that unnamed "proponents of separate registration contend that the nature of authorship embodied in the computer program code is substantially different from the authorship typically embodied in computer screens." It also acknowledged that everyone agrees that the same screen displays can be generated by many, very different codes. Even accepting those points, however, the Copyright Office stated that "this does not alter the fact [sic] that the computer program code and screen displays are integrally related and ordinarily form a single work." Those who disagree with this policy, the Office said, "must concede that the program code and screens are conceptually a single work."

Finally, the Office stated that it did not intend to address the concern of some witnesses (those witnesses from the Computer Society, for example) that the copyright registration rules applicable to screen displays should specify the boundaries or scope of the copyright to provide greater guidance to users. The Copyright Office stated that it was "sympathetic to users who may have difficulty in determining the scope of copyright," but it considered their problem not to be its responsibility. The Office said that it will attempt "to keep out of the public record any frivolous, unsound, or otherwise unjustified" copyright registrations, but that is all it is prepared to do to "assist the public and the courts." They will simply have to make their own way in the world.

The Office's June 1988 policy statement needs analysis on two levels: first, in terms of what it expressly says; second, and more important, in terms of what it fails to address. The first, or facial, level of analysis need not long detain us, and we may proceed directly to that. The second level of analysis requires further foundation to be laid, however. It calls for a description of what undiscussed alternatives were or are available for protecting screen displays, how practical they are in terms of ease of implementation, and what the likely consequences are of adopting one or the other of the different choices.

A facial critique. There are serious problems with the Office's June 1988 policy statement, initially even at a linguistic level. Most of it is purely conclusory, some of its assertions are on their face not credible, and what passes for its basic rationale is at best metaphysical incantation. The policy statement may appropriately be subjected first to a critique at the facial level, to push aside the verbal debris. Then criticism can continue at a more fundamental level, in terms of the policy issues that are not addressed but nonetheless must be considered in any legitimate analysis.

First, the imagined public benefit that the Office perceives in a single copyright application and single $10 fee, instead of two applications and $20, is trivial. Second, the alleged confusion engendered by having screen displays registered separately from code exists only in the minds of the authors of the Copyright Office's statement. The Copyright Office does not explain how separate registrations of the two things will increase confusion, and the assertion is implausible.

Third, the alleged dangers of multiple copyright infringement actions and multiple claims for statutory damages are figments of someone's imagination. For one thing, pirates of screen displays do not copy (and in the reported cases have not copied) the code. If they did copy the code, it would be easier just to sue them for copying the code; that they do not copy the code is why we must address the screen-display copyright infringement problem. Moreover, the stated fear of excessive, multiple statutory damages based on multiple suits and registrations ignores the fact that the amount of statutory damages to award to the plaintiff is discretionary with the court. There is no reason to assume that federal judges are any less able than the Copyright Office in assessing the equities of this type of situation, and that they are incapable of sensibly exercising the discretion that Congress confided in them. In short, the calculation of benefits and harms that the Office uses to justify its policy against separate registration of screen displays is hopelessly flawed.

The "fact that the computer program code and screen displays are integrally related and ordinarily form a single work" is the supposed conceptual rationale for fusing code and screen displays, for copyright purposes. But in what sense is that a fact? It is not a fact; it is a conclusion; and the conclusion is mere ipse dixit.

Even worse, what does it mean? In what sense are code and display integrally related, and what does integrally related mean, if anything? The diskettes on which products such as 1-2-3 or dBase are sold ordinarily contain both working code (for doing the work) and display code (for generating the displays) on the same diskettes. The display code on such diskettes generates the displays in question here. Those are facts, whether you call it integral relation or something else. But there is no rational connection between those facts and the conclusion that there should be a single copyright covering the code and the visual display, even if integral relation is a suitable epithet or characterization to apply to them. Incidentally, "integrally related" is neither a term defined (even implicitly) in the copyright statute nor a term of art used in custom-honored copyright case law and jargon.

By the same token, the analysis is not furthered by asserting that the code and display form a single work. That is again a conclusion, not a factual observation. It does not assist us in determining whether public policy is better served by addressing the copyright in a screen display separately from or together with the copyright in the related computer program.

Moreover, the assertion is also arbitrary, as it is for any related works or composite work. For example, the music associated with "Greensleeves" has been used for other works, such as the carol, "What Child Is This?" By the same token, many of the songs in Shakespeare's plays have been given several different musical settings; "O Mistress Mine" is an example. The same sort of thing could be said of the different artists' illustrations of *Alice in Wonderland* and the Pooh books. Some reasonable observers would agree that the words and music, or books and pictures, form a single work; others would disagree; any such assertion is necessarily arbitrary. Even more important, one may ask only in vain how to solve any practical business or legal problem by attempting to determine whether the music and words, or pictures and words, form a single work.

In short, the Office provides no analysis here at all. It offers only an unexplained conclusion, smothered in language that obscures any factual content. Indeed, only one piece of fact appears in all of the Copyright Office's formal explanation of its policy: "The Copyright Office benefits by having a simplified administrative process." Viewers of programs such as "Yes, Minister" will doubtless appreciate the true rationale of the Office's decision.

In the next issue I turn to a further critique of the Copyright Office's decision. What are the range of alternatives in addressing protection of screen displays and related aspects of user interfaces? How do different options relate to the policy concerns described earlier? To give readers a road map: I conclude that protecting screen displays by means of existing copyright law is more feasible than trying to develop a more ideal system. I also conclude that separate registration of screen displays: a) makes more sense than any other approach, and b) probably can provide appropriate incentives and encouragements to creation without stifling software progress.

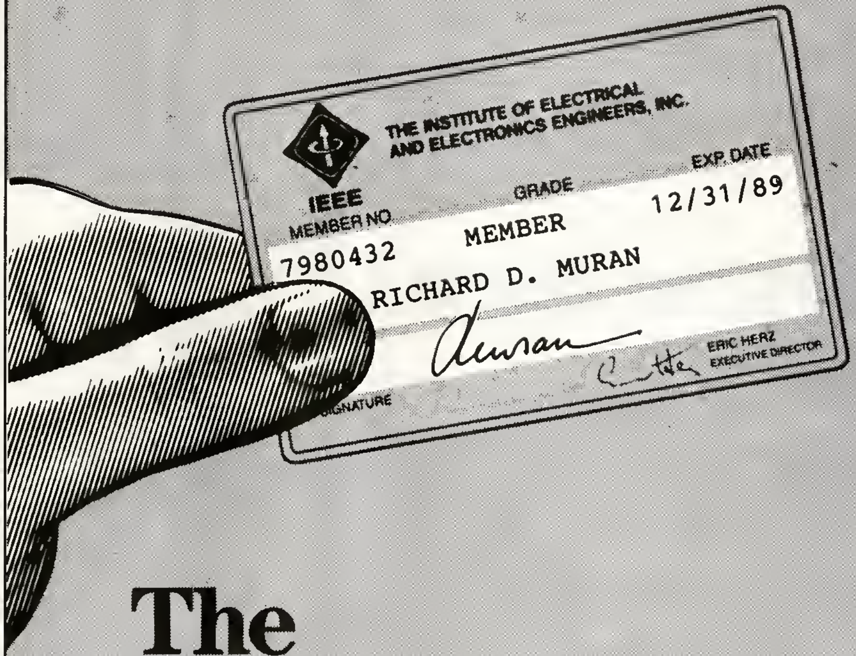
References

1. Whelan Associates, Inc. *v.* Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986), *cert. denied,* 107 S. Ct. 877 (1987).
2. Manufacturing Technologies, Inc. *v.* CAMS, Inc., 706 F. Supp. 984 (D. Conn. 1989).
3. R.H. Stern, Micro Law, *IEEE Micro,* Vol. 9, No. 4, Aug. 1989, pp. 7-10, 92-94.
4. *IEEE Micro,* Vol. 6, No. 6, Dec. 1986, p. 75.
5. *IEEE Micro,* Vol. 9, No. 3, June 1989, p. 84.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 171 **Medium** 172 **High** 173



The Ultimate High-Performance Vehicle

IEEE operates worldwide through 36 technical societies, organized in hundreds of local chapters. We can put you in touch with a chapter near you.

Discover the advantages of belonging to the world's largest professional technical society—The Institute of Electrical and Electronics Engineers, Inc.

FOR A *FREE* MEMBERSHIP INFORMATION KIT USE THIS COUPON.

Name ____________________

Title ____________ Phone ( ) ____________

Firm ____________________

Address ____________________

City ____________ State/Country ____________ Postal Code ____________

MAIL TO: IEEE MEMBERSHIP DEVELOPMENT
The Institute of Electrical and Electronics Engineers, Inc.
445 Hoes Lane, P.O. Box 1331
Piscataway, N.J. 08855-1331, USA
(201) 562-5524

IEEE

Department

Micro News

Micro News features information of interest to professionals in the microcomputer/ microprocessor industry. Send information for inclusion one month before cover date to Managing Editor, IEEE Micro, *PO Box 3014, Los Alamitos, CA 90720-1264.*

Microprocessors challenge supercomputers

Ware Myers
Contributing Editor

"The latest microprocessors are within striking range of supercomputer performance," Eugene D. Brooks III, staff physicist in the Computational Physics Division at Lawrence Livermore National Laboratory, asserted in a recent interview.

These microprocessors promise about half the performance of a Cray 1S, providing that the scalar code being run "hits cache well." (The Cray 1S is the traditional measuring unit of supercomputer performance.) Furthermore, supercomputer users will soon find it economic to rethink and recode applications to run on parallel machines built from microprocessors, according to Brooks' analysis.

Brooks has pursued parallel-processing and computational physics research at Lawrence Livermore in California since 1983. Prior to that, as a research assistant at the California Institute of Technology, he contributed to the development of the Hypercube, a large distributed machine based on commercial microprocessor technology. His PhD in physics is from Caltech.

The Intel i860 64-bit RISC-based microprocessor represents the level of performance that Brooks has in mind. Containing 1,000,000 transistors, the architecture of this device embraces many supercomputer concepts, such as memory management, instruction cache, data cache, pipelined architecture, one operation per clock cycle, and fully pipelined floating-point operations. In addition to the 40-MHz i860, other high-performance microprocessors exist in the marketplace.

The memory-bandwidth problem

A limitation on the performance of the recent microprocessors is their ability to get data from main memory into the computing units, Brooks pointed out. For applications that are generally satisfied by data in cache memory, performance is very high. In applications that have to go out to main memory for data rather frequently, throughput slows down compared to supercomputers because main-memory bandwidth of these microprocessors is much less than that of supercomputers.

The latest microprocessors use page-mode dynamic RAMs that speed up memory access a little bit, says Brooks, but not substantially. Supercomputers contain very large memory subsystems that deliver one word per clock to the processor. Designers usually construct these subsystems by interleaving large numbers of relatively fast static RAM chips and overlapping the accesses to get high bandwidth.

"Can you do this right on the memory chip itself instead of stacking up 64 memory chips glued together with MSI [medium-scale integration] logic chips?" Brooks asked himself. "Yes, all you have to do is put that logic right on the memory chip itself. In this way you can overcome the very poor main-memory bandwidth of these fast microprocessors."

Supercomputer on a chip

In terms of single-processor hardware, Brooks expects to see "the exact architecture of the classic supercomputer stuck down on one chip that holds the cache, a floating-point unit, a CPU, and an interface designed to handle memory chips with internal interleaving in two to five years." These systems will literally be the equivalent of the classic supercomputer, both in terms of scalar performance and vector performance—as soon as someone straightens out this last problem, the memory bandwidth.

These systems should be mounted on little boards, maybe a little larger than one's hand. "They will be someone's personal micro," Brooks said. Of course, they will be 100 times cheaper than the supercomputers they will be replacing.

"If you look at the applications we are actually running at the laboratory, you find that 90 percent of them are 5- to 10-minute runs. People could happily run such applications on this latest group of microprocessors, and run them 10 or 100 times cheaper," Brooks added. "The number of applications right now that actually use all of the processors [in a multiprocessor supercomputer], use all of main memory, and run in a fully chained vector mode at top speed are remarkably few."

Parallel architectures

"The real supercomputers of the future will be composed of hundreds of these boards, arranged so that they can talk to each other," Brooks went on. "Right now there are two thrust areas in this regard. The first is the high-performance equivalent of an E-mail network. Here the programs running on the various processors send messages to each other, with the programmer writing explicit code to send and receive messages. In the second all of the processors are wired to share each other's memory, and a given processor can read or write another processor's memory at will."

The Intel iPSC2 is an example of message-passing architecture. The BBN Butterfly represents a shared-memory design.

The only real difference between these two architectures lies in the higher communications capability of the shared-memory approach. "On message-passing machines the user program constructs a message and hands it off to a system call on the local processor. This processor wires the communication to the remote processor, which hands the message to a system call, which finally hands it to the user program," Brooks outlined.

"The communications performance on message-passing arrangements is not dominated by the hardware, but by all those layers of software. It is this extra systems-software overhead that makes message-passing machines poor performers when it comes to communications. On shared-memory machines the construction, dispatching, and responses to messages are handled directly in the hardware supporting the shared-memory paradigm."

Brooks argued that a parallel system with 100 or so of these powerful microprocessors will provide performance about 100 times beyond that of the classic supercomputer "quite soon." In fact, "the new microprocessors can do it now for applications that don't need the main-memory bandwidth and that can run well out of cache. But eventually they will do that even for vectorized codes."

Rethinking the code

These parallel processors still must have code generated for them. Brooks remains dubious about converting "dusty Fortran decks" to parallel code automatically. "People who tell you they have an automatic parallelizing compiler for even a four-processor system are selling you snake oil," he smiled. "We have tested a lot of these supposedly automatic compilers, and they just don't work for anything real."

He was not talking about vectorizing compilers. They do work. He referred to the compilers that look at the whole program and have to figure it out.

Still, a good many problems in the physical sciences are inherently parallel in the first place. In the past they have been turned into single-thread programs by sweat and tears. In the future they will be written in parallel fashion from the beginning.

"We are learning that we have to write explicitly parallel code for these new machines," Brooks reported. "There are parallel programming languages and packages used to enhance Fortran, C, and other languages that allow us to do a good job."

Of course, rethinking and recoding will take time and cost money. "If a new machine is only twice as fast, nobody is willing to spend a lot of time rethinking or rewriting the code. There is no leverage," Brooks emphasized. "If the new machines are about the same cost, but 100 times faster than a present-day supercomputer, suddenly people in droves will want to reprogram their applications for them. There will be enormous leverage."

Still, this revolution won't arrive next month. "No, but it might be here in a couple of years," Brooks forecast. "There has been a lot of experience with parallel machines over the last five years. People doing academic work in this area have had the vision to see the future. There are many new journals reporting results. We have a base to draw upon. Now suddenly with the latest crop of microprocessors, the leverage is there. It is no longer an academic exercise. We can still win the war against Amdahl's law."

[Eugene Brooks' views represented in this interview are his personal opinions and do not reflect those of the United States Government or the University of California.—Ed.]

Shockley and the transistor

William Shockley, 79-year-old co-inventor of the transistor, Nobel laureate in physics, and early advocate of California's Silicon Valley, died of cancer in August this year.

Highly controversial for his views on social science, Shockley nevertheless holds an enviable place in history for his part in inventing the first semiconductor device. This device became the seminal innovation of the current electronic age, as most modern devices contain its linear descendants. Former employees at Shockley Semiconductor in Palo Alto, California, went on to the invent the integrated circuit and the microprocessor.

While at Bell Laboratories, Shockley, John Bardeen, and Walter Brattain combined their knowledge and experience to find a solid-state semiconductor replacement for the classic vacuum tube amplifier invented by Lee De Forest in 1907.

After a year of failing to find a replacement, the team achieved success. On December 16, 1947, Brattain used a wedge-shaped piece of plastic to press two narrowly spaced, parallel-line gold contacts less than an inch apart against a block of germanium. When an electric signal was introduced to the germanium through the wires, the device amplified the signal. Another employee, John Pierce, called the device a transistor because it transferred current across a resistor.

The year of repeated failures prompted Shockley to say later, "A basic truth that the history of the invention of the transistor reveals is that the foundations of transistor electronics were created by making errors and following hunches that failed to give what was expected. Pure inspiration contributed less to progress than did perseverance and the willingness to try again." He later called the process "Creative-Failure Methodology."

Shockley purchased the transistor patent for $25,000 when he left Bell Labs in 1956 and later founded his own company with major financing from Beckman Instruments. His inability to manage engineers and physicists led to eight of his best researchers breaking away two years later to found Fairchild Semiconductor. The group included Robert Noyce, who went on to coinvent the integrated circuit, and Gordon Moore, who later invented the microprocessor. Noyce and Moore later founded Intel Corp.

For almost two generations, most Silicon Valley start-up companies could trace lineage back to Fairchild and before that to Shockley Semiconductor. In fact, an industry genealogy chart of that family tree decorated walls in area offices for years.

Shockley lectured in engineering science at Stanford University in California for many years. Transistor coinventor Brattain died in 1987. Bardeen, who went on to win a second Nobel Prize for his work on superconductivity, is professor emeritus at the University of Illinois.

Posix to speak new language

Recognizing the need for international open systems, the IEEE Computer Society Posix committee recently formed a study group to guide development of emerging open systems standards. The group meets this October 16-20 in Brussels to present a preliminary draft of a reference model and internationalization guidelines for discussion. The European Commission hosts the meeting.

The Posix, or Portable Operating System Interface for Computer Environments, family of standards originated in response to demand for applications portability with the rapid evolution of computer technology. Posix makes user applications independent of the underlying computer hardware and system software.

Non-US end users of applications currently must make their systems speak their language and comply with their cultural and business conventions. Formats of date and time, use of local language characters, and numeric formats typify the difficulties these users face.

The newly formed study group plans to provide the framework for developing standardized interfaces to reduce the effort and associated costs for localizing Posix systems. Its work will incorporate work already done by the Posix working group in Japan, the /usr/group Technical Committee (an international group of Unix-compatible system users), X/Open, and other industry groups.

The initial Posix standard is both an ISO Draft International Standard and a US government Federal Information Processing Standard (FIPS 151).

Micro Bits

North American microcomputer users can reach the popular French **Minitel** information network with a local call when using Minitel's free terminal software. With modem parameters set at 1200-8-N-1, call BBS number 1 (800) 999-6163 and enter "Minitel" at login prompt to order the software.

Representatives of ANSI, CEN, and CENELEC standards bodies met recently to **prevent barriers to free trade** as Europe moves toward creating a single market by 1992. They believe cooperative coordination of standards, certification, and testing will improve day-to-day business for US and European industries.

GE Superabrasives and Asahi Diamond Industrial Co. Ltd. of Tokyo joined forces to develop and produce diamond materials using chemical vapor deposition. **CVD diamond**'s thermal conductivity properties create a variety of electronics application opportunities.

Boston's **Computer Museum** shows SIGGraph 89 computer art through January 4, 1990. Displays include two-dimensional works, giant moving sculpture, interactive environments, and animated items.

NIST announces a computerized database of commercial products successfully implementing the **Open Systems Interconnection** standards. The experimental Osinet lets vendors test each other's products for interoperability. Current vendors include DEC, Hewlett-Packard, IBM, NCR, Retix, Touch Communications, Unisys, The Wollongong Group, and Xerox.

An **HDTV** standards working group plans to develop technical specifications and engineering standards for US/Canadian TV and film producers offering videotape for later broadcast or satellite transmission. Contact F.M. Remley, Jr., at (313) 763-7479 for data on the Society of Motion Picture and Television Engineers project.

Micro Standards returns

In response to repeated requests to reinstate the Micro Standards column, Editor-in-Chief Joe Hootman recently announced the appointment of Carl Warren to *IEEE Micro*'s Editorial Board to fill this opening. In addition, Warren will begin a new bus department, On the Edge, beginning this December.

Warren, a McDonnell-Douglas Space Systems Company senior scientist/senior engineer, is a senior voting member of the IEEE Microcomputer Standards Committee governing council. In addition, he serves as publicity chair for the IEEE Standards Activities Board of the Computer Society.

Previous assignments included work as a design engineer and as a senior editor for *EDN* and *Mini-Micro Systems* magazines. In addition, he has authored numerous technical articles and books and served as technical advisor to the Buscon, Comdex, Interface, and NCC trade shows and the *War Games* movie. Warren, who is currently pursuing an MSEE degree at the University of California, Long Beach, holds degrees in industrial technology and chemistry.

Microprocessors in education

Conference co-chairs William G. Vogt and Marlin H. Mickle announce that special sessions on microprocessors in education will take place at the 21st Annual Pittsburgh Conference on Modeling and Simulation next May 3-4.

Vogt and Mickle will review previously unpublished papers in these special sessions—if received by January 31, 1990—for possible publication in a special issue of *IEEE Micro*. This issue will address the concerns of educators who transmit microprocessor information to students.

Submissions should describe significant contributions that add to the knowledge in a particular area or describe the progress of research currently being conducted.

The conference proceedings will carry every accepted paper. Direct questions about the conference or submission requirements to either cochair, Modeling and Simulation Conference, 348 Benedum Engineering Hall, University of Pittsburgh, Pittsburgh, PA 15261.

ICS Electronics Corporation, 2185 Old Oakland Road, San Jose, CA 95131; (408) 432-9009; fax (408) 943-1745.

AT&T's Unix System Videotape Library is a series of videocassettes covering all levels of the system. Selected topics include fundamentals (basic, intermediate, and advanced), shell command language, C language, and system administration.

AT&T Videotape Library, Jacksonville, FL; (800) 554-6400 ext. 7140; from $450.

/usr/group, the International Association of Unix Systems Users, states that its 1990 products directory to be published this December will be free to general members. Associate members pay $25 and the public, $50.

/usr/group, 2901 Tasman Drive, Suite 201, Santa Clara, CA 95054; (408) 986-8840.

The *TMOS Power MOSFET Selector Guide* from Motorola provides tables of linear ICs that are available to drive power MOSFETs. The SG56/D guidebook describes high-energy E-FETs, Sense FETs, and standard devices.

Motorola Inc., Literature Distribution Center, PO Box 20924, Phoenix, AZ 85063; (800) 521-6274; free.

The 698-page *Proceedings of the Second International Workshop on Signal Processing of HDTV* includes 66 papers from SMPTE, NHK, Sony, CCETT, MIT, BBC, Bellcore, and Philips.

North-Holland, PO Box 1991, 1000 BZ Amsterdam, The Netherlands; Dfl. 310.00; or PO Box 882, Madison Square, New York, NY 10159; $163.25.

Be a winner!

IEEE Software invites entries for the third annual Gordon Bell Prize until December 31, 1989. Interested researchers should enter their applications of parallel processing to scientific and engineering problems for consideration of two $1,000 awards. The competition centers on three categories: performance, price/performance, and compiler parallelization. Entrants should submit a three- to four-page executive summary describing their work.

Prize sponsor and former National Science Foundation division director Gordon Bell plans to announce winning entries in *IEEE Software* next spring.

Submit entries to and request rules from 1989 Gordon Bell Prize, c/o *IEEE Software*, PO Box 3014, Los Alamitos, CA 90720-1264.

Current literature

Practical applications for neural networks, tested architectures, and designs appear in the 64-page quarterly, *Journal of Neural Network Computing*. Ongoing columns will discuss applications in image analysis, visual processing, speech analysis/recognition, knowledge processing, control systems, robotics, biological neural systems, and machine learning.

Auerbach Publishers, One Penn Plaza, New York, NY 10019; (212) 971-5000; fax (212) 971-5081; $145/yr.

Accounts of the most frequently encountered dilemmas confronting the 488 bus user appear in *Extending the IEEE 488 Bus*. The 28-page booklet also describes mix and match solutions for expansion, extension, and control.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 183 **Medium** 184 **High** 185

IEEE MICRO

1990 Editorial Calendar

FEBRUARY

Hot Chips issue

Details of the hottest microchips on the market today! Explore the architectures of National's embedded processor, Sun Microsystem's Sparc 64-bit system and graphics accelerator, Intergraph's Clipper, Intel's i486, the MIPS RISC with floating-point unit, Texas Instrument's floating-point chip, and more. Update your knowledge of Motorola's 88000, Weitek's Sparc EP, and gallium-arsenide processors.

Ad closing date: January 1

AUGUST

Communications and buses

Making RISCs work together; choosing the best software and interfaces; adding PC slaves and MS-DOS, memory technology, LANs, message-passing techniques...

Submit manuscripts by: 1-15-90
Ad closing date: July 1

APRIL

Special Far East issue

The latest technology from Japan, Korea, Taiwan, and the Pacific Basin and current TRON Project offerings.

Ad closing date: March 1

OCTOBER

Digital Signal Processing

Is it the end of the supercomputer era? Comparison of supercomputer Linpack benchmark price/performance rankings with the latest 32-bit floating-point DSP chips makes serious analysts contemplate the future of the supercomputer industry.

Submit manuscripts by: 3-15-90
Ad closing date: September 1

JUNE

General Interest Issue

Working with ASICs, operating systems, bioengineering, or other micro-related topics? We're interested in publishing an account of your efforts.

Submit manuscripts by: 11-15-89
Ad closing date: May 1

DECEMBER

Special European and Near East issue

With 1992 just around the corner, US companies feel the pressure to meet new economic rules to compete for sales in Europe. What are European companies doing to meet US plans for common-market penetration?

Submit manuscripts by: 5-15-90
Ad closing date: November 1

IEEE Micro helps designers and users of microprocessor and microcomputer systems explore the latest technologies to achieve business and research objectives. Feature articles in *IEEE Micro* reflect original works relating to the design, performance, or application of microprocessors and microcomputers.
All manuscripts are subject to a peer-review process consistent with professional-level technical publications. *IEEE Micro* is a bimonthly publication of the IEEE Computer Society.

Advertising information: Contact Marian Tibayan, Advertising Department, IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

*Articles may change. Please contact editor to confirm.

Micro Review

Richard Mateosian
Hitachi America, Ltd.
(415) 244-7456
Fax (415) 583-4207

Leftovers

In my December 1988 column I mentioned an excellent program editor for the PC called Brief, by Underware. Unfortunately, I did not have information on the publisher available when I prepared that column, and I have since been asked about it. Underware, 84 Gainsborough Street #103W, Boston, MA 02115, wrote Brief. Solution Systems, 541 Main Street #410, South Weymouth, MA 02190; (800) 821-2492; publishes it.

In my August 1989 column, I made some comments about Wingz that were seen by at least one reader as somewhat more negative than I had intended them to be. To clarify, I have a generally positive opinion of Wingz, but there are, in my opinion, a few rough edges to the total package. *(See Letters, p. 7, this issue.—Ed.)*

Freshly prepared

Born to Code in C, Herb Schildt (Osborne McGraw Hill, Berkeley, Calif., 1989, 525 pp., $28.95)

This is a really interesting book for anyone interested in programming. In the tradition of the still unsurpassed *Software Tools* by Kernighan and Plauger (Addison-Wesley, 1976), the book offers real, useful, well-written programs, illuminated by informed commentary. In this case the author provides an additional device: Interspersed among the pages of the nine chapters are 14 one-page thumbnail sketches, based upon interviews conducted with prominent C programmers.

While both the main text and the programmer profiles prove interesting, they are basically unrelated, which weakens the book. It would have been better simply to make the set of profiles into its own chapter. Then some of the contrasts and parallels between individual programmers might have been developed. As it is, with the profiles isolated from one another and none especially deep, they don't add much to the book. For those who keep track of such things, all 14 of the programmers selected for inclusion are men.

Here are some of the insights provided by these programmers:

"Keep rewriting each chunk of code until it's unabashedly elegant and readable—no matter what its supposed importance. Think about portability and efficiency from the start, but make no sacrifices for either until testing shows you must."—P. J. Plauger

"If you understand how your compiler generates code, you will be better able to write highly efficient, optimized programs."—Tom Green

"First make it work. Then make it work right. Finally, make it work fast!"—Donald Killen

"Experiment. Take chances. Don't be afraid to crash your machine. Push yourself to your limits of understanding. Write code. Rewrite code. Rewrite your rewrites."—Jeff Betts

"C's greater freedom of expression makes it easier for a good programmer to write a program that is isomorphic to his or her conceptual solution."—Robert Ward

"Use the power of C, don't abuse it."—Robert Jervis

"Use the language conservatively; don't become enamored with the tricky things you can do."—Brad Silverberg

"If you don't thoroughly understand the problem or cannot think of a clear solution, no programming style or language can help you."—Bjarne Stroustrup

"Clarity and organization are the keys to creating great programs. Become a great programmer and you will get the most out of C."—Ralph Ryan

"A great program performs its task unerringly."—Jack Blevins

"Great programs all share one common feature: they are intuitively easy to use by their target users."—Richard Schwartz

"It is a good idea to mimic modern software-engineering practices, such as modular design. Also, you should use full-function prototyping."—Jesper Boelsmand

"When I need to do a detailed design, I draw the BNF structure of it. This not only forces me to think through the program carefully, but gives me the outline of it as well."—Thomas Plum

"What makes C special is its suitability to modular programming."—Dennis Saunders

Insights of this sort are interesting, but the book must stand or fall on the contents of the other 511 pages. Reading lots of code, as one of the profiled programmers points out, is important to developing facility in programming. This book gives you the opportunity to read a large body of real code that was written with readability in mind and to do so with the author looking over your shoulder and pointing out the key elements and explaining the design decisions.

The book begins with an interpreter for a subset of the ANSI C language called Little C. If you've used computer languages but are a little hazy on how they work, this sort of material can be very helpful. A simple interpreter is much easier to understand than a simple compiler, and once you have it working, it is much easier to experiment with. An understanding of the workings of an interpreter for a language can greatly enhance your understanding of how to use the language. In the absence of this sort of concrete connection between expression and action, it is easy to form misconceptions about how certain language features work or what they mean.

The next chapter deals with icon-based interfaces. Many people use visual operating system interfaces, like that of the Macintosh computers. These interfaces are easy to use but hard to understand, and books like *Inside Macintosh* don't help much. The material that Schildt presents here is pretty simple, but it does give readers an idea of how these interfaces work. You need an IBM PC with some sort of graphics adapter to run the code.

Another issue deals with terminate-and-stay-resident functions, again in the PC-DOS environment. TSR functions, as their name implies, remain in memory ready to run while other programs are being used. Usually a special keystroke, called a hot key, activates the TSR, which then performs its intended task and returns control to the running program. DOS contains features, not all of which have been documented, to support the implementation of TSR functions. Schildt explains how TSR functions work, documents the key undocumented support features in DOS, and develops a substantial TSR application.

The first three chapters are really the most helpful ones, since the subject matter is both valuable and not commonly available. The next three chapters also present useful, but more mundane programs: a multitasking kernel, a screen editor, and a database system. In the final three chapters, Schildt returns to the offbeat, dealing with custom fonts, animation, mouse interfacing, and printer control.

Something ought to be said here about Schildt's programming style, since this is, after all, a book meant to teach the art of programming in C. Unfortunately, the fact is that if you are looking for interesting algorithms, cleverly concise code, or any other sorts of highlights, you'll be disappointed. Like most real programs, this code contains nothing to be skimmed. It is competent and neatly formatted. It follows the usual practice of using lowercase characters exclusively, except for defined constants, which are all uppercase. Comments are sometimes, but not always, complete sentences, beginning with an initial capital letter and ending with a period. Indentation and bracketing follow a consistent and widely used convention. White space is consciously used for clarity, but sometimes the code gets a little dense.

In summary, I like this book, and if you are interested in programming, I think that you'll like it too.

The NeXT Book, Bruce F. Webster (Addison-Wesley, Reading, Mass., 1989, 416 pp., $22.95)

This is not your usual "Gee whiz" book about a new gizmo. The Next computer system is an excellent piece of work, and this book does give a good overview of it. As early reviewers often do, Webster has worked closely with Next. He has been given access to equipment and information that were not readily available, so that he is understandably grateful to his benefactors. He does sometimes gush embarrassingly about them. Nonetheless, this book is not hype. It is technically competent and informative.

I spent some time playing with a Next computer system shortly before reading this book, so I was able to read it with a concrete picture in mind. The book clarified many things that had puzzled me. It would certainly be a useful piece of documentation to be shipped with Next systems.

The Next computer system is housed in a 1-foot cubic black box, which contains four 96-pin Euro DIN connector slots connected by a modified Nubus. One of these is devoted to the processor board, and the others are available for other uses. Next provides a bus interface chip for developers of plug-in cards. The box contains a 3.5-inch optical disk drive and can accommodate another internal 3.5-inch drive—either a hard drive (SCSI) or another optical drive. The processor board contains a 68030 CPU, a 68882 FPU, a 56001 DSP, 8 to 16 Mbytes of dynamic RAM, some video RAM and static RAM, and a custom 12-channel DMA processor. The display, including keyboard, mouse, and sound equipment, forms a separate unit. Another unit contains the Next laser printer, a bare-bones device that uses a Canon engine and relies upon the 68030 on the processor board for the intelligence that is usually built into other laser printers.

It would certainly be a useful piece of documentation to be shipped with Next systems.

On top of this standard set of hardware sits the Unix-like Mach operating system and a set of tools for the user interface. In addition, the Write Now word processor, the Mathematica database server, a dictionary, a book of quotations, and the complete works of Shakespeare are bundled with the system.

Webster's book methodically describes all of these elements in a top-down manner. Early chapters seem like baby talk, but precise and accurate baby talk that indicates that the author knows more than he is saying. And in due time further layers of detail are revealed. By the end of the book, most sophisticated readers will feel quite comfortable working in the Next environment without further documentation. What more can you ask?

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 177 **Medium** 178 **High** 179

Advertiser/Product Index

	RS #	Page #
BOARDS		
Enhancement board	57-58	91
Fastboards	91	94
Motherboard	86	93
STD-compatible board	89	94
System board	80	93
CHIPS		
Data interface	64	92
DRAMs	23-25	88
SRAMs	14-22, 29-33	88, 89
Video RAM	26-28, 34	89
DATA ACQUISITION		
Conversion system	67	92
DATA COMMUNICATIONS		
Design service bureau	63	92
MEMORY/STORAGE EQUIP.		
Memory module	10-13	88
Portable disk drive	44	90
RECRUITMENT		
Classified advertising	—	91
Technical professional	—	1
SOFTWARE		
Connectivity software	46	90
Fortran compiler	84	93
RISC CAE software	56	91
Simulation package	—	C.IV
SYSTEMS		
ASIC design-automation tool	35-43	89-90
Computer	81, 83, 85, 88, 90	93, 94
Coprocessor	53	91
Database management system	52	91
Graphics controller	49-51	91
Operating system/ multitasking executive	47-48	90
PCB design package	59-60	92
PCB pattern library	62	92
Power platform	82	93
Processor module	87	94
RISC	54-55	91
TEST & MEASUREMENT EQUIP.		
Integrated signal-noise analysis tool	61	92
Multiprocessing architecture	45	90
Symbolic and numerical package	65-66	92

CACI Products Company	Cover IV
IEEE Computer Society Membership	19
IEEE Membership Development	79
Intel Corp.	1
Classified Advertising	91

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northern California and Pacific Northwest: Roy McDonald Assoc. Inc., 5915 Hollis St., Emeryville, CA 94608; (415) 653-2122.
Jim Olsen, P.O. Box 696, Hillsboro, OR 97123; (503) 640-2011.

Southern California and Mountain States: Richard C. Faust Co., 24050 Madison St., Suite 100, Torrance, CA 90505; (213) 373-9604.

Midwest: The Kingwill Company, 4433 W. Touhy Ave., Suite 540, Lincolnwood, IL 60091; (312) 675-5755.

East Coast: Atlantic Representative Group, 349 Maple Place, Keyport, NJ 07735; (201) 739-1444.

New England: Arpin Associates, P.O. Box 6444, Holliston, MA 01746; (508) 429-8907.

Europe: Heinz J. Görgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G.); phone: (0 21 53) 8 99 88; telex 841(17)2153310=HJG tlx d.

Southwest/Southeast: Publications Office, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

Moving?

Address changes: Please notify us 4 weeks in advance

ATTACH LABEL HERE

- This address change notice will apply to all IEEE publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

Name (Please Print)

New Address

City

State/Country Zip

Mail to:
IEEE Micro, Circulation Dept., 10662 Los Vaqueros Cir., Los Alamitos, CA 90720-2578

Department

New Products

Marlin H. Mickle
University of Pittsburgh

Send announcements of new microcomputer and microprocessor products, and products for review, to Managing Editor, IEEE Micro, *PO Box 3014, Los Alamitos, CA 90720-1264.*

Refreshing your memory

One million characters in a module

Two single in-line memory modules (SIMMs) that come in 3.5-inch-long, 1-Mbyte configurations allow users to upgrade system memory without changing PCBs. The company estimates that a 1-Mbyte SIMM can store one million alphanumeric characters.

The 8-bit KMM581000 consists of eight KM41C1000 DRAMs in 20-pin SOJ packages. The KMM591000 contains an additional DRAM for parity checking. Both SIMMs come with 80-ns to 100-ns row-access times, 8-ms refresh times, and 30-pin dual-in-line packaging. **Samsung Semiconductor; from $185.95 (100s).**

100-ns 581000
Reader Service Number 10
80-ns 581000
Reader Service Number 11
100-ns 591000
Reader Service Number 12
80-ns 591000
Reader Service Number 13

SRAMs reach 20-ns speeds

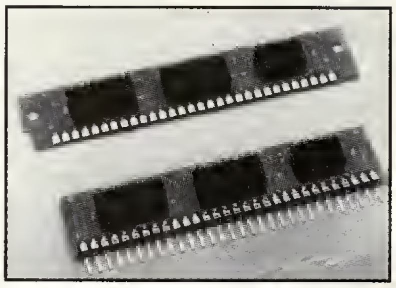
The MCM6208 and MCM6209 64-Kbyte × 4-bit fast SRAMs feature 1.0-micrometer HCMOS technology and 20-ns access times. A sister device, the 256-Kbyte × 1-bit MCM6207, also operates at 20 ns.

The company also plans 1.0-micrometer versions of its 16-Kbyte × 4-bit MCM6288 and MCM6290 devices, as well as the 64-Kbyte × 1-bit MCM6287—all with 15-ns access times.

Upgraded MCM4180, MCM62350, and MCM62351 application-specific memory devices of 16-Kbyte density now feature 20-ns access times. **Motorola.**

4180 **Reader Service Number 14**
6207 **Reader Service Number 15**
6208 **Reader Service Number 16**
6209 **Reader Service Number 17**
6287 **Reader Service Number 18**
6288 **Reader Service Number 19**
6290 **Reader Service Number 20**
62350 **Reader Service Number 21**
62351 **Reader Service Number 22**

Packaging accents density



Three AK59256 dynamic RAM modules come mounted on printed wiring boards in SIMM or SIP configurations.

According to the company, AK59256 DRAM modules have six times the density as those packaged in DIP form. The modules contain two 256-Kbyte × 4-bit CMOS DRAMs in SOJ packaging and one 256-Kbyte × 1-bit DRAM in PLCC packaging.

In the lower 8 bits of this 9-bit-word module organization, each tristate-data I/O pin connnects to two 256-Kbyte × 4-bit DRAM bidirectional data pins. The separately controlled ninth bit can provide parity. **Accutek (pricing varies in quantities from 5 to 100,000 units).**

Reader Service Number 25

Sleight-of-hand controllers

Two VLSI DRAM controllers update slower DRAMs by making them appear faster to the system. The KS84C21 and the KS84C22 also interface between system-memory array and the microprocessor to eliminate a number of logic devices. Interleaving provides access to DRAM array without fast-page, static-column, or middle-mode DRAMs.

The KS84C21 and KS84C22 support 256-Kbyte and 1-Mbit DRAMS; the KS84C22 also supports 4-Mbit devices. **Samsung.**

84C21 **Reader Service Number 23**
84C22 **Reader Service Number 24**

Big VRAM fires graphics

The CMOS, 1-Mbit Video RAM features access times of 80 ns and flash-write, fast-page-mode, and write-per-bit operations. The M5M482128 configuration combines a 128-Kbyte × 8-bit dynamic RAM parallel-access, memory-array port with a 256-Kbyte × 8-bit, serial-access-memory port.

The M5M442256 version combines a 256-Kbyte × 4-bit DRAM port with a 512-Kbyte × 4-bit SAM port.

The M5M482128 features a bidirectional SAM port with split read/write capabilities, block write, and logical operations. VRAM applications include laser printers and imaging displays for PCs and workstations.

The M5M482128 comes in 40-pin, small-outline, J-lead (SOJ) packaging, while the M5M442256 comes in either 28-pin SOJ or 28-pin zig-zag, in-line (ZIP) packages. **Mitsubishi Electronics America; $62 (M5M482128) (100s); $57.50 (M5M442256) (100s).**

482128
Reader Service Number 26
442256 (SOJs)
Reader Service Number 27
442256 (ZIPs)
Reader Service Number 28

SRAMs expand the bandwidth

The 1-Kbyte × 8-bit IDT7050 and the 2-Kbyte × 8-bit IDT7052 Four-port SRAMs allow simultaneous, asynchronous access of memory locations from any one of four ports to improve system speed. Interfacing four IDT7052s with 32-bit processors or DMA devices results in an effective bus bandwidth of 640 Mbytes/s.

The expandable, 25-ns SRAMs come in 108-pin plastic and ceramic PGA packaging and 132-pin plastic and ceramic quad flat packs. Applications include multiprocessing and DSP. **Integrated Device Technology; $212.50 (IDT7050, 108-pin ceramic PGAs) (100s); $297.50 (IDT7052, 108-pin ceramic PGAs).**

7050 **Reader Service Number 29**
7052 **Reader Service Number 30**

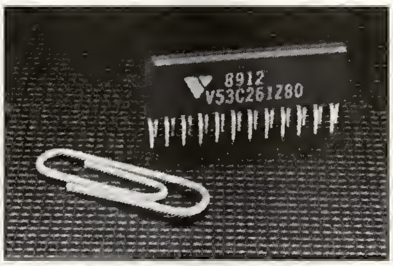
SRAMs feature TTL interfaces

Three SRAMs with transistor-transistor-logic I/O interfaces provide 12-ns address access times and 16-Kbit densities. The SSM6116, SSM6168, and SSM6170, fabricated with proprietary BiCMOS process technology, provide conventional pin-out and function compatibilities.

The SSM6116 offers 2-Kbyte × 8-bit organization with common data I/O functions. The SSM6168 and SSM6170 provide 4-Kbyte × 4-bit organization. The SSM6170 also offers an output-enable function to control the tristated bus. The SRAMs come in a variety of packaging. **Saratoga Semiconductor; from $21.70 (SSM6116); from $20.80 (SSM6168); from $20.80 (SSM6170); (all 1,000s).**

SSM6116
Reader Service Number 31
SSM6168
Reader Service Number 32
SSM6170
Reader Service Number 33

VRAM boasts quick access times



The V53C261 CMOS video RAM comes in 24-pin, plastic zig-zag, inline packaging.

The V53C261 VRAM clocks 256 4-bit words at a serial rate of 33 MHz with an address access-time of 80 ns. Real-time read-transfer operations can occur during row-address changes. Other features of the 64-Kbyte × 4-bit device include a bit-mask write function, fast-page-mode access, RAS-only and CAS-before-RAS refresh, and bidirectional data transfer between serial and DRAM ports. **Vitelic; $14.63 (100s).**

Reader Service Number 34

ASIC design

Integrated system aids ASIC design

The Fastrack design-automation tool in concert with various cell libraries provides IC design for commercial and military signal-processing, control, and power applications.

The digital version offers timing and area optimization, hierarchical schematic capture, logic and fault simulation, static-timing analysis, and standard-cell placing and routing.

The bipolar analog version offers a statistical process database, menu-driven interfaces, hierarchical schematic capture, coupled electrical and physical-design programs, and layout-versus-schematic checking.

The toolset allows automatic back-annotation of layout modifications and parasitics for resimulation. **Harris Semiconductor; from $20,000 (digital, front end); $60,000 (analog, software only).**

Digital
Reader Service Number 35
Analog
Reader Service Number 36

Supercells come to the PC

Zyp-AT Design Tools incorporate Supercell VLSI peripherals as well as standard cells for schematic capture, netlist conversion, simulation, and test-program development. Supercells contain a pretested functioning lock to reduce development time and increase chip complexity on IBM PC ATs.

Consisting of timers, peripheral interfaces, interrupt controllers, and clock generators, the Supercell library requires a 40-Mbyte hard disk, one parallel or two serial ports, and 1 Mbyte of extended memory per 1,000 gates for simulation operations. Zyp-AT tools run with Microsoft Windows or DOS with DOS extenders. **Zymos Corporation.**

Reader Service Number 37

New Products

Tool boosts design

The Logic Assistant design-automation tool integrates design planning, capture, testability, and verification functions into one environment. Designers can proceed from initial specification through partitioning and estimation analysis to high-level functions with graphic feedback for critical path analysis and fault grading.

Designers can also gain feedback on trade-offs between gate array and cell-based designs, power consumption, chip size, and package selection. The Logic Assistant is a component of the V8 ASIC Design Platform. **VLSI Technology; $15,000 (option in nonstandard configurations); no cost (in some standard configurations).**

Reader Service Number 38

CAD ASIC tools run on Sun

LSI Logic has developed two design tools that run on all-ASIC Sun Sparcstation 1 and 4/330 platforms.

Silicon Builder automatically produces physical block layouts at the chip level.

Design Builder—used with the Silicon Integrator toolset—estimates chip partitioning and die size and automatically compiles memory cells. The package also synthesizes finite state and register-transfer machines. **LSI Logic.**

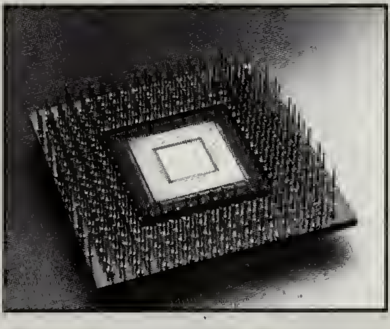
Silicon **Reader Service Number 39**
Builder **Service Number 40**

Interface enhances VHDL

The VHDL interface allows designers to specify designs in the VHSIC hardware description language at behavioral, dataflow, and structural levels for simulation on Ikos 2300/2900 systems. The interface consists of a VHDL analyzer and compiler and the C++ modeling language. Dataflow and structural descriptions translate into simulated gate-level netlists. **Ikos Systems.**

Reader Service Number 43

Mix-and-match design tools



The OACS environment supports this bipolar, semicustom circuit assembled in tape-automated bonding, PGA packaging.

The Open Architecture CAD System hosts third-party design tools on Apollo and Sun workstations. Vendors of accommodated tools include Mentor Graphics, Valid Logic Systems, Cadence Design Systems, Synopsys, and Gateway Design Automation. Users can customize designs with such functions as logic synthesis, multichip/multilevel logic simulation, and digital timing analysis. The Electronic Data Interchange Format environment supports both CMOS and bipolar ASIC technologies. **Motorola; about $10,000 (license per node, low-end configurations); about $100,000 (fully configured system licenses).**

(Low-end)
Reader Service Number 41
(Licenses)
Reader Service Number 42

Have data, will travel

The Traveldisk portable disk drive stores 10 to 200 Mbytes of data and plugs into IBM PC XT/ATs, 80386s, laptops, and Macintosh computers. The company claims ruggedness for the 3 × 5 × 7-inch steel-case subsystem, which weighs 3.5 pounds. Bus-extender cards, laptop connectors, or Mac SCSI cables complete the package. **Tradewinds Peripherals.**

Reader Service Number 44

Take a RISC on critical timing

The TC2000 RISC for time-critical applications boasts a multiprocessing architecture that expands from eight to 504 processors. Function cards that include the 88000 microprocessor, 32 Mbytes of on-board DRAM, and optional I/O capabilities fit on one circuit board. The company reports performance rates of 19 Dhrystone MIPS, 13 million Whetstones/s, and 20 Mflops.

The system supports two operating systems at the same time. While some processors run the Psos+m real-time executive, others can use the Nx Unix-based 4.3 BSD system. **BBN Advanced Computers; $350,000 (base model with graphical development and debugging tools).**

Reader Service Number 45

Package bridges DEC and IBM systems

Ezbridge connectivity software for the DEC RISC Ultrix family of systems accesses CAD/CAM packages on IBM mainframe/midrange systems and X.25 networks. Plug-and-play components residing on DEC-supported communications boards transfer large binary graphics files to and from workstations at high speeds. Users can also build peer-to-peer applications to integrate data from both IBM and DEC systems using the LU6.2 protocol.
Systems Strategies.

Reader Service Number 46

Vrtx32 for the 80960

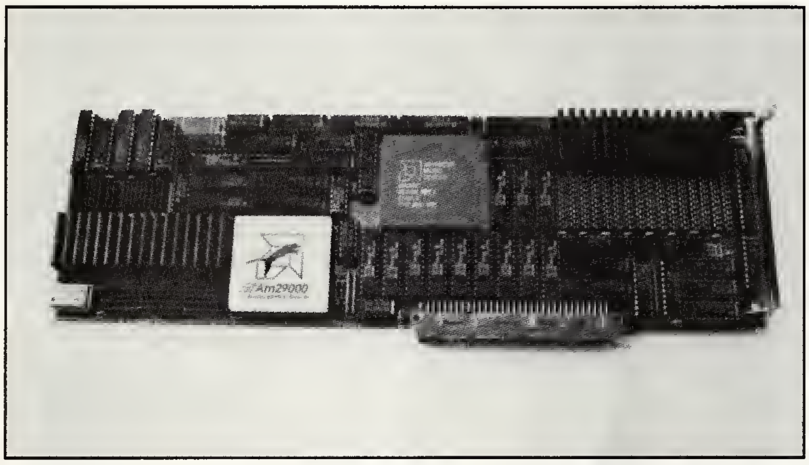
The Vrtx32/960 operating system/multitasking executive for real-time embedded applications supports Intel's 80960 RISC. The kernel features deterministic, fixed-cost system calls independent of the number of tasks, queues, or other system objects. Designers can program applications on Sun, PC, or VAX hosts.

The company plans product availability in December 1989. **Ready Systems; from $4,810 (Vrtx32/960, individual license); $7,800 (debugger/monitor).**

Vrtx32
Reader Service Number 47
Debugger
Reader Service Number 48

RISC and RISC-related products

Coprocessor for the Mac II



The Nusuper coprocessor features a 32-bit Am29000 RISC CPU, a 50-MHz system clock, and a peak performance of 25 VAX-equivalent MIPS.

The Nusuper system provides 3 Mbytes of cache plus direct access to 32 Mbytes of DRAM on the Nubus. Macintosh, PC, Sun, or VAX hosts can compile code to run on Nusuper under the Macintosh operating system. The company supplies the Metaware C compiler and Macintosh OS runtime interface, as well as the assembler, linker, and debugger. A bus master can directly access graphics and other I/O cards. **Yarc Systems; $3,995 (basic system).**

Reader Service Number 53

Unix, 80-MIPS RISC runs with graphics

The Tp881V VME-based multiprocessor configuration—combined with the company's Tp-Ix Unix operating system—allows multiple users to simultaneously access the system kernel.

The Tp-AGCV graphics controller provides a 2,048 × 2,048 × 8-frame buffer with a graphics processor and supports up to 32 hardware windows. **Tadpole Technology.**

881V **Reader Service Number 49**
Ix **Reader Service Number 50**
AGCV **Reader Service Number 51**

DEC supports RDMS

The Ingres relational database-management system and 4GL tools now run on Ultrix-based Decsystem 5400 and 5800 computers and Decstation 2100 workstations. A multi-server, ANSI-compliant architecture provides portability of applications and transparent data access across divergent systems. **Relational Technology; from $5,000 (single-user configuration).**

Reader Service Number 52

Faster 88000s abound

Motorola has announced volume production of 88000 RISCs that perform at speeds up to 25 MHz at 21 MIPS. The company reports benchmark performance measurements of 48,387 Dhrystones and 25,000 Whetstones per second. **Motorola; $697 (25-MHz 88100); $875 (25-MHz 88200).**

88100
Reader Service Number 54
88200
Reader Service Number 55

CAE meets Ultrix

The Workview Series II CAE software for ASIC and analog design runs on RISC Decstations 3100 in the Ultrix operating-system environment and on Unix-based DEC and Sun workstations. The package provides a framework for integrating proprietary and third-party CAE/CAD tools into one system.

The toolset employs Decwindows interfacing, multitasking, and networking. **Viewlogic; from $10,000.**

Reader Service Number 56

Convert your PC into a RISC

The Risclink transputer-based enhancement board for the IBM PS/2 provides parallel processing and RISC functioning over the Micro Channel bus.

The 32-bit accelerator connects clock, communications, and power between on-card transputer modules and the PC.

T414, T425, and T800 32-bit modules feature 10 MIPS and 4 Mbytes of RAM with 25-MHz clock rates. The T800 also provides 64-bit, floating-point support.

A starter kit consists of one card and a 1-Mbyte T800 module, an assembler, and the C programming language. **Levco; $799 (Risclink); $1,495 (kit).**

Board **Reader Service Number 57**
Kit **Reader Service Number 58**

CLASSIFIED AD

RATES: $5.00 per line, $50 minimum charge (up to ten lines). Average five typeset words per line, eight lines per column inch. Add $10 for box number. Send copy at least one month prior to publication to: **Heidi Rex or Marian Tibayan, Classified Advertising, *IEEE Micro*, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.**

SENIOR PRODUCT DESIGN ENGINEER

Manufacturer of industrial equipment in Greensboro, NC has immediate need for Software/Hardware Development Engineer. Responsibilities include product design and development of multi-processor, real time operating system for state-of-the-art, highly advanced machinery. Assembly languages 8086/286/386 and C required. BSEE, 3-7 yrs. exp. with industrial background pref. Competitive salary, attractive benefits and clear growth opportunity being offered.

Resumes to: P.O. Box 93
Landmark Sq.
Stamford, CT 06901

PCB design

Roll-your-own commands

The DC/CAD enhanced PCB design package allows users to create their own commands with a built-in programming language. Language constructs include If/Endif, While/Endwhile, Exit, Wait, and Getkey.

Improved Gerber output enhances photo-plotting capabilities. The package, which now supports 96 apertures, features a command history that allows users to employ one-keystroke, repeat commands.

A digitizer interface supports Autodesk's ADI interface for most commonly used tablets. **Design Computation; from $495 (DC/CAD-level-2 users eligible for free updates within one year of purchase).**

Reader Service Number 59

From micros to mainframes and back again

Workstation- and mainframe-based PCB design tools can bilaterally trade data directly with P-CAD Master Designer II and Associate Designer PCB design packages for the PC.

Users can start a PCB design using P-CAD on the PC, place components and route traces on workstation Professional Cadam Prance software, and integrate finished designs with a packaging system designed on the mainframe-based Cadam Interactive Design package.

Design data transfers between applications can take place electronically. **Cadam, Inc.**

Reader Service Number 60

Tool analyzes signal noise

The Allegro PCB Engineering System now includes a set of integrated signal-noise-analysis tools.

Designers can analyze board designs by estimating the delay and distortion characteristics of high-speed signals.

The system also provides thermal and reliability analysis and calculation of wire delays and parasitics.

Other board analyses include reflections, crosstalk, and ohmic loss.

Design engineers and PCB designers can interactively view analysis results, identify electrical errors, implement design modifications, and evaluate the effect of changes. **Valid Logic Systems; from $7,500 (depending upon software configuration).**

Reader Service Number 61

Library supports surface-mount technology

The Tango-PCB/SMT Plus pattern library, based on James Blankenhorn's *SMT Land Pattern Handbook*, includes 116 vendor-specific patterns such as quad flat pack, inductor, and Mil Spec.

Many of the International SMT Association-endorsed patterns have been refined and tested under volume-production conditions.

Designers can add additional patterns to existing Tango SMT libraries. **Accel Technologies; $149 (includes handbook and library on disk).**

Reader Service Number 62

Bureau provides CAD routing service

Automated Systems, Inc. claims overnight completion of 24 × 24-inch boards with 1,500 components. ASI design service bureaus—equipped with a Prance design system running on an IBM-4300 series mainframe and a Prance GT on the Decstation 3100 workstation—can interface to a number of systems.

The bureau accepts an ASCII-placed netlist via magnetic tape, floppy disks, or data over dial-up telephone lines. Designers can preroute critical signals that require specific routing geometries.

ASI can generate artwork and fabricate prototypes or supply specified tooling. **Automated Systems, Inc.**

Reader Service Number 63

Chip set supports workstations

The Supernet Fiber Distributed Data Interface (FDDI) add-in board supports 100-Mbps data rates and allows connection of 500 network stations.

The VLSI five-chip set addresses physical protocol and media-access control FDDI LAN standards. (The physical media-dependent standard defines optical data links, cables, and connectors.) Users can address the station-management function standard in software. **Advanced Micro Devices.**

Reader Service Number 64

Mainframe math on MS-DOS

The PC Macsyma symbolic and numerical package runs on 80386-based PCs using the MS-DOS operating system. Featuring the same capabilities as the workstation and mainframe versions, the package solves differential and integral equations, computes Laplace and Fourier transforms, and automates vector and tensor calculus. **Symbolics; $2,900 each; $510 each (10 copies, certain academic institutions).**

Commercial
Reader Service Number 65
Academic
Reader Service Number 66

Transfer tape from mainframes to PCs

The nine-track Reformatter conversion system contains a 0.5-inch tape drive, controller card, and software for conversion, tape backup, or restoration. Controllers for IBM PCs and PS/2s support file/data/program transfers between mainframes and PCs. The software runs with MS-DOS, Xenix, and Unix operating systems. **Microtech Conversion Systems; from $3,000.**

Reader Service Number 67

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 180 **Medium** 181 **High** 182

Product Summary

Marlin H. Mickle
University of Pittsburgh

For more information, circle the appropriate Reader Service Number on the Reader Service Card.

Manufacturer	Model	Comments	R.S. No.
80X86, i486-related machines			
Mylex Corp.	MWS386-25 system board	Intel 80386-based board operates at 25 MHz and supports DOS, Unix, and OS/2 operating systems. Zero wait states result from 64 Kbytes of 25-ns SRAM and 80-ns SIMM DRAM chips. An AT-standard bus and eight memory slots support LANs, file servers, workstations, and CAD/CAM systems. From under $1,300.	80
Entourage Computer Corp.	316XMC, 320MC PCs	PS/2-compatible PCs feature the Micro Channel architecture and 3.8-inch profiles. The Intel 80396SX-based, 16-MHz 316XMC and the 80386-based, 20-MHz 320 MC come with 1 Mbyte of RAM and a 40-Mbyte, SCSI hard disk. $2,695 (316XMC); $3, 895 (320MC).	81
IBM	PS/2 486/25 power platform	Upgrade platform for the PS/2 Model 70-A21 replaces the 80386 board with the 25-MHz, 1.2-million transistor i486 and retains compatibility with IBM PC, OS/2, and PS/2 operating systems or 386 software. RISC design, an 8-Kbyte, internal cache memory, and an on-chip math coprocessor complete the package. $3,995.	82
Systems Integration Associates	SIA 486 computer series	Intel i486-based computer family features 64 Kbytes of static cache memory plus the i486's 8-Kbyte, built-in cache. AT- and EISA-bus versions both include a 4-Mbyte main memory, a 150-Mbyte main disk, a 1.2-Mbyte floppy, and serial or parallel ports. The SIA 486/25 and /25E turn in 25-MHz performances; the SIA 486/33 and 33/E clock at 33 MHz.	83
Language Processors, Inc.	Fortran compiler	Compiler for the Sun 386i workstation running Sun OS Version 4.0 implements ANSI X3.9-1978 Standard and Mil-Std-1753 and complies with X Open. VAX Fortran and Fortan-66 extensions help import applications. Enhancements to Fortran-77 include Do While and End Do statements, an Include directive, extended Integer and Logical data types, and octal and hexadecimal contants. Also includes Codewatch debugging tool to test source-level programs. $995.	84
Systems Integration Associates	SIA 386/33 computer	Zero-wait-state, 64-Kbyte cache-memory computer runs at 32.64 MHz according to *Byte* Lab five-test suite. (Scoring is equivalent to an IBM AT at 1.0.) A 4-Mbyte disk cache supports database and network applications.	85
US Integrated Technologies	OEM-2LP motherboard	Available in 16-, 20-, and 25-MHz versions, 80286-based AT PC incorporates VGA color graphics, a floppy disk controller and hard-disk interface, one parallel and two serial ports, and up to 4 Mbytes of DRAM. The surface-mount-technology system also uses Phoenix BIOS.	86

Product Summary

Manufacturer	Model	Comments	R.S. No.
Xycom, Inc.	XVME-683 processor module	IBM PC AT-based computer packaged on two VMEbus cards features 1 or 4 Mbytes of dual-ported memory, SCSI and floppy disk controllers, and a color graphics controller. Users can chose optional 80387 or Weitek math coprocessors. PC AT bus allows direct communcations between cards without using the VMEbus. A battery-backed clock and watchdog timer support industrial applications.	87
Micro Express	ME 286-12SL and ME 286-16SL PCs	Two 80286-based PCs in 16.5 × 4 × 15.25-inch cases include a monochrome graphics card and 14-inch flat-screen monitor. Also provided are 640 Kbytes of RAM, a choice of 5.25- or 3.5-inch floppy disk drives, and serial and parallel ports. Other features include five expansion slots and a socket for an 80287 coprocessor. $799 (12-MHz 286-12SL); $1,199 (286-16SL).	88
Winsystems, Inc.	MM-286AT-20-2M board	STD-compatible board runs at 20 MHz and supports OS/2, MS-DOS, Xenix, and PC-DOS operating systems. Users can configure the 2-Mbyte board with up to 2 Mbytes of parity DRAM, 128 Kbytes of EPROM, and two DMA or 15 interrupt channels. From $1,995.	89
Fortron/Source	386 Netset 333 computer	Zero-wait-state, 33-MHz system for CAD/CAM/CAE, file serving, and and multiuser applications comes with 1 Mbyte of memory and a 64-Kbyte cache of 15-ns SRAM. Also includes choice of 5.25- or 3.5-inch floppy disk drives, one parallel or two serial ports, and an ESDI controller. $4,900 (not including monitor and hard drive).	90
AST Research	486/25 and 386/33 Fastboards	Industry Standard Architecture-based upgrades for 25- and 33-MHz Premium 386 desktops offer i486 or 386 CPUs. According to the company, the 33-MHz 386 provides a 33-percent performance gain for the 25-MHz Premium 386. $2,995 (486/25 upgrade for the 386/33); $3,695 (486/25 upgrade for the 386/25); $2,395 (386/33 upgrade for the 386/25).	91

Micro View

continued from p. 96

Microcode, in part, disguises the nitty-gritty hardware details of the implementation and maps them onto a standardized instruction set. Random logic, PLAs, and state machines can also disguise details. But microcode allows the greatest capability for producing "machine-level" instructions that are not closely related to native hardware functions.

Cache memory, while primarily an implementation feature rather than an architectural one, does have some characteristics that can be visible to software. Ideally, the existence of a cache should be transparent to the software. If data or instructions reside in the cache, the processor can retrieve them more quickly, but the function remains the same. However, cache memory has some subtle architectural implications. Some memory locations (such as memory-mapped I/O) must not be cached. Most systems provide some way for software to control which address ranges are cached and which are not, as well as some mechanism to globally enable or disable the cache.

In general, only operating system software uses these functions. Thus, they represent a separate, and less critical, level of architectural compatibility. Processor designers can change functions that are used exclusively by system software without affecting application programs. Thus they worry less about changing these functions than about general architectural characteristics that affect all software.

As with cache memory, only operating system software typically uses memory management functions. However, unlike the cache control functions, the interactions between system software and memory management hardware are complex. Furthermore, if designers expose the memory management model to user-level software, changing the model can have significant effects on applications programs. The major offender in this regard is Intel's 80X86 architecture. The 8086 uses base registers, which are accessible by user programs, as part of the memory management scheme. The application programmer must explicitly deal with the resulting segmented memory space. The segment registers function differently on a 286 (in protected mode) than on an 8086. So 8086 application software is not upward compatible to the 286 unless the 286 runs in "real mode," effectively disabling all memory management functions.

Thus, while memory management functions do not strictly form part of the basic processor architecture, changing them can be painful for software designers.

Sparc typifies an architecture that was initially defined without a memory management unit. Sun has its own MMU design, which it has patented. It is not TLB-based (translation look-aside buffer) and requires a considerable amount of static RAM and discrete logic. (Sun makes the patent available for licensing, but no chips yet implement Sun's MMU scheme.) Seeing the need for a standard, Sun eventually introduced its "reference" MMU specification for VLSI (very large scale integration) implementation. Sun's latest workstations use a proprietary ASIC (application-specific IC) implementation of Sun's original MMU scheme, however. Future Sun clones will likely use the "reference" MMU chips that are becoming available.

As a result, while Sparc application programs should enjoy binary compatibility across a variety of hardware implementations, operating-system code will have to be changed for each MMU. Sun heavily modified its current version of Sun OS to isolate the MMU-dependent code and make it easier to change the MMU model. In earlier Unix versions, changing the MMU was a large project, affecting many parts of the system software.

Architectural issues

The truly architectural issues center on the instruction set and the register file. While we see many differences among instruction sets, within the range of architectures we consider here (32-bit microprocessors), few differences assume significance other than the distinction between RISC (reduced instruction-set computing) and CISC (complex instruction-set computing) styles. To be sure, the Motorola 68020 instruction set differs from the 80386 instruction set, but these differences are relatively unimportant. Similarly, the instruction sets of Sparc, MIPS, Motorola 88000, and Am29000 processors do not differ much. Each has some weak points, such as Sparc's lack of an integer multiply instruction (rather than just a multiply step). But these differences rarely become deciding factors in selecting a processor.

I do not mean to say that architectural issues are unimportant. Countless pages have been written—and will be written—on the advantages and disadvantages of various architectural features. Given two equally good implementations, architecture is certainly important in making a selection.

The architecture also affects the implementation possibilities. Indeed, this is the key reason for RISC architectures, and makes them easier to build than CISC designs. A minimum Sparc CPU requires less than 20,000 gates (about 80,000 transistors). It achieves performance well in excess of that achieved by the Intel 386 with about 300,000 transistors (although the 386 includes an MMU, while the Sparc chip does not). As a result, we will see Sparc implementations in gallium arsenide long before we see GaAs 386 implementations.

Two key differences separate RISC and CISC architectures. The differences have nothing to do with "reduced" instruction sets versus "complex" instruction sets, but nonetheless we commonly associate them with these categories. Because most CISC architectures were defined when the number of transistors per chip was relatively small, architects made some trade-offs that are no longer optimal.

First, most CISC architectures offer fairly small register sets—usually no more than 16 registers, and often no more than eight. All RISC architectures include 32 or more registers, which significantly reduces the number of memory load and store operations that must be performed. No straightforward way exists to extend the register set of CISC architectures while retaining binary compatibility.

Second, most CISC architectures use an accumulator-based or two-operand instruction format, which destroys one of the operands by overwriting it with the result of the calculation. This approach requires additional instructions to copy the operand to another register if it is needed after the calculation.

These two characteristics give today's RISC architectures an advantage, but a new CISC architecture could use them as well. Nevertheless, the lack of these features is a permanent disadvantage for today's CISC architectures.

Implications

Why bother with these fine distinctions? They become important because all future compatible chips will be stuck with the architecture, while implementations can change. Selecting a microprocessor is often a long-term decision. Once a company adopts a particular processor for a product line, the inertia of the software that has been developed, expertise gained, and vendor relationships make it painful to switch architectures later on. Designers can remedy defects in today's implementation in future chips, but they find defects in architecture much more difficult to correct.

Architectures are, of course, abstract entities, and physical systems must be built with available implementations of that architecture. Thus, the characteristics of current implementations are important. If, however, we select a processor to form the basis of what is hoped to be a long-lived product line, we may willingly live with a suboptimal implementation if the architecture is sound. We might also have a good reason to think that future implementations will be better. For many designers, the question is not which chip seems best today. Rather, designers ask, "Which processor family will have the best chip (for my application) in three years, when my product goes into volume production. And, what will be available for the generation after that?"

Designers should also ask, "How long will the recently established RISC architectures evolve while maintaining binary compatibility?" For example, IBM's second-generation RT will require recompilation of existing programs. Some observers feel RISC architects inevitably will want to add more complex addressing modes and other "CISC" functions to increase performance. This type of evolution will occur primarily in an upward-compatible fashion, but some designs may deviate from binary compatibility to allow a significant performance improvement.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

186 Low **187** Medium **188** High

Department

Micro View

Michael Slater
Micro Design Resources Inc.
(415) 494-2677
mslater@cup.portal.com

Distinguishing architectures from implementations

[With this issue, IEEE Micro *welcomes Michael Slater as he assumes responsibility for this column. The editor and publisher of* Microprocessor Report, *an industry technical newsletter, Slater authored the textbook,* Microprocessor-Based Design. *He also organizes the annual Microprocessor Forum symposium, the technical conference for microprocessor chip and system designers. He welcomes your comments and suggestions for the column.—Ed.]*

The field of microprocessor-based design continues to evolve at a rapid pace, as it has for nearly 20 years now. In the past few years a flurry of new microprocessor architectures, bus structures, interface standards, operating systems, and design tools has emerged, producing many opportunities but also much confusion.

Selecting architectures and design approaches for microprocessor-based systems is not an exact science. The tradeoffs are complex, and the diversity of options available in commercial products makes the decision task all the more difficult. Well-informed buyers talk to a variety of vendors and perform their own analysis of each product's strengths and weaknesses for their application. Unfortunately, since each vendor tends to view the world in a way that is most favorable to its product, such research projects often produce as much confusion as clarity.

In this bimonthly column I plan to clarify some of the key issues in microprocessor-based hardware design. In this first column, I wrestle with the various meanings of *architecture* and *implementation*, explore the implications of the distinction, and discuss its importance to the microprocessor selection process.

What is architecture?

Computer scientists use the term *architecture* in various ways. One usage, which I will try to follow, refers to *instruction set* architecture. In essence, this usage includes those aspects of the microprocessor design that are visible to the software. Many completely different hardware designs may follow the same architecture, allowing them to run the same software. The differences among such designs are *implementation* differences.

The machine-language instruction set forms the core of a processor's "personality." Chip designers must take care with changes to the instruction set if architectural compatibility is to be maintained. Designers maintain *upward* compatibility (the ability to execute old code on a new machine) by including an existing instruction set as a subset of a new instruction set and using previously undefined opcodes to add new instructions. *Downward* compatibility (the ability to run code for a new machine on an older one) can be provided if older machines trap on undefined opcodes. The trap handler on the old processor then emulates the new instructions.

Implementations can vary greatly while retaining architectural compatibility. The term *microarchitecture* refers to the detailed implementation, and has an entirely different meaning than *architecture* as used above. *Architecture* is often used in either way, so we must be very careful in interpreting comments about it. In addition, different levels of architectural compatibility exist.

Levels of visibility

Software functions completely unaware of many implementation details, except that the speed of execution may be affected. A processor with a 32-bit architecture may have only a 16-bit adder with microcode or other sequencing logic to perform a 32-bit add in two steps. For that matter, it could have a single-bit adder and perform the addition in 32 steps. Similarly, the width of the data bus is not an architectural issue, as long as the hardware will provide the sequencing necessary for data transfers of any size defined by the architecture. Other common examples of architecturally irrelevant implementation details include the use of a barrel shifter to speed up multibit shift operations and the various algorithms that can be used to implement multiplication and division in hardware.

continued on p. 94

0272-1732/89/1000-0096$01.00 © 1989 IEEE

IEEE COMPUTER SOCIETY

A member society of the Institute of Electrical and Electronics Engineers, Inc.

Computer Society Offices

Headquarters Office

1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone (202) 371-0101
Telex: 7108250437 IEEE COMPSO
Fax: (202)728-9614

Publications Office

10662 Los Vaqueros Cir.
P.O. Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information: (714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office

13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office

Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 408-3118

Use the Reader Service Card to obtain information on:

- Membership application—student #203, others #202
- Perodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Standards working groups list #195
- Compmail+ international electronic mail/database brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures—student/regular #193
- Student scholarship information #192
- Awards description/nomination forms #198
- Volunteer leaders/staff directory #196
- IEEE senior member application #204

Purpose

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

Membership

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others seriously interested in the computer field.

Publications and Activities

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and new products.

Periodicals. The society publishes six magazines and four research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

European Office

Payments for Computer Society membership and publication orders are accepted by checks in Belgian, British, German, Swiss, or US currency. Checks in US funds must be drawn on a US bank. Payment may also be made by American Express, Eurocard, MasterCard, or Visa credit cards.

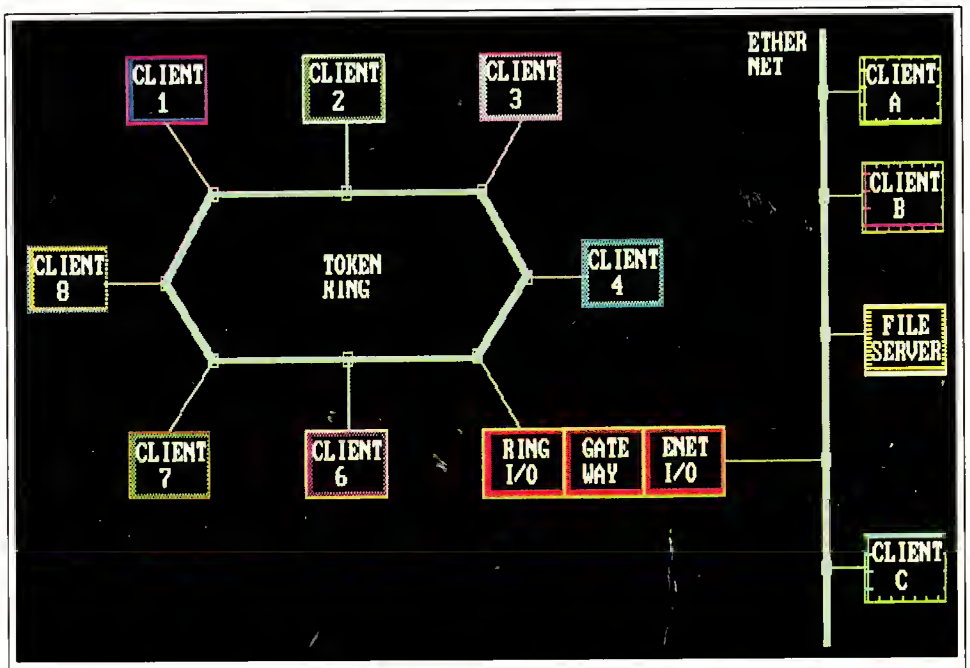
Asian Office

Payments for Computer Society membership and publication orders are accepted by checks in Japanese or US currency. Checks in US funds must be drawn on a US bank. Payment may also be made by electronic fund transfer to the Bank of Tokyo, Akasaka Branch, Toza acct. 0767956; the credit receiver is the IEEE Computer Society Headquarters Office. Payment may also be made by American Express, Eurocard, MasterCard, or Visa credit cards.

Ombudsman

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

NEW FOR LAN PLANNERS



Realistic simulation of your Local Area Network, quick results --no programming

SIMLAN II now predicts performance of LAN's

Free trial and, if you act now, free training

SIMLAN II uses simulation to predict your LAN performance. You simply describe your LAN and workload.

Animated simulation follows immediately--no programming delays.

Easy-to-understand results

You get an animated picture of your LAN. System bottlenecks and changing levels of utilization are apparent.

You can model clients, servers, bridges and gateways.

Your reports show system statistics such as transfer times, information throughput, and message loss. Client, server, and gateway statistics show queue lengths, waiting times, and blocking.

Computers with SIMLAN II

SIMLAN II is available for most PC's and Workstations.

Your network simulated

You can predict the performance of any LAN. Industry standard protocols such as Ethernet, Token Ring, Token Bus, and FDDI are built-in. Others can be modeled.

You can easily study the effect of changing parameters or even protocols.

Seeing your LAN animated increases everyone's understanding of its operation and builds confidence in your results.

Free trial information

The free trial contains everything you need to try SIMLAN II™on your computer. For a limited time we also include free training **--no cost, no obligation.**

Call Paul Gorman at (619) 457-9681, Fax (619) 457-1184. In Europe, call Richard Eve on (01) 528-7980, Fax (01) 528-7988.

Free trial offer

See for yourself how SIMLAN II quickly answers LAN performance questions.

Limited offer--Act now for free training.

☐Send information on your Special University Offer.

Name ______

Organization ______

Address ______

City ______ State ______ Zip ______

Telephone ______ Fax ______

Computer ______ Operating System ______

IEEE MICRO

Return to:
CACI Products Company
3344 North Torrey Pines Court
La Jolla, California 92037
Call Paul Gorman at (619) 457-9681.
Fax (619) 457-1184.

In Europe:
CACI Products Division
Regent House, 89 Kingsway
London WC2B 6RH, United Kingdom
Call Richard Eve on (01) 528-7980.
Fax (01) 528-7988.